DEVELOPING A DEVICE FOR AUTOMATIC MONITORING OF ROLLING ELEMENT BEARING CONDITIONS

A Thesis
Presented to
The Academic Faculty

By

Niklas B. Tritschler

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering and the University of Stuttgart

Georgia Institute of Technology
August 2019

DEVELOPING A DEVICE FOR AUTOMATIC MONITORING OF ROLLING ELEMENT BEARING CONDITIONS

Approved by:

Dr. Kurfess, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Sawodny
Institute of System Dynamics
University of Stuttgart

Dr. Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Tarín
Institute of System Dynamics
University of Stuttgart

Date Approved: July 02, 2019

ACKNOWLEDGEMENTS

I would like to especially thank my advisor Dr. Thomas Kurfess for his guidance during this research. Our regularly meetings and discussions as well as his excellent support and input helped me to get a deep understanding of this field of research and to create new ideas and approaches.

I would also like to thank Andrew Dugenske for his support and help during our discussions about the characteristics of the hardware components and the digital architecture.

A great thanks is also dedicated to the Ford personnel. We had great discussions and I am grateful for their sponsoring and the opportunity to test my device on their machines.

I would like to thank all my lab colleagues, which supported me on different topics. We had a great and productive atmosphere.

Finally, I would like to thank my parents for their support during all my years of study. Without their help, I would not have enjoyed so many opportunities and my studies would have been less successful.

Niklas B. Tritschler

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ADC	Analog-To-Digital Converter
API	Application Programming Interface
BW	Bandwidth
CBM	Condition-Based Maintenance
CNC	Computerized Numerical Control
d_B	Rolling Element Diameter
d_P	Pitch Diameter
DFT	Discrete Fourier Transform
f	Frequency
f_s	Sampling Frequency
f_N	Nyquist Frequency
FFT	Fast Fourier Transform
HMI	Human Machine Interface
I2C	Inter-Integrated Circuit
IoT	Internet of Things
MCU	Microcontroller Unit

MQTT	Message Queuing Telemetry Transport
N	Total Number of Samples
OEE	Overall Equipment Effectiveness
P_v	Peak Value
PWM	Pulse Width Modulation
QoS	Quality of Service
RMS	Root-Mean-Square
RPM	Rotations Per Minute
RTC	Real Time Clock
SBC	Single-Board Computer
TTL	Transistor-Transistor Logic
Δt	Time of each Sample
UART	Universal Asynchronous Receiver-Transmitter
x_i	Vibration Signal of one Sample
θ	Contact Angle
$\omega_{Spindle}$	Rotational Speed of the Spindle

SUMMARY

In most instances, rotating machines have a unique vibration signature that relates to their health status. Therefore, vibration analysis is a powerful tool for predictive maintenance. This is especially true for bearings that are a frequent cause of machine breakdown. Presently, bearing analysis of many machines results in significant cost and complexity due to a large amount of vibration data that must be analyzed. The purpose of this thesis is to develop a vibration analysis system that locally collects vibration data, analyzes it automatically and provides feedback as to the bearing condition. This system shall be internet enabled and embedded into an Internet of Things architecture. Thereby, location-independent control of the system and its results is possible.

Based on the characteristics of rolling element defects, a vibration measurement box is developed that consists of the single-board computer (SBC) BeagleBone Black, the microcontroller (MCU) Teensy 3.2 and the accelerometer ADXL203EB. The accelerometer measures the vibration of the spindle and is mounted on it with a magnet. The vibration signal is sampled with the MCU and is transferred to the SBC via a UART connection. This SBC has a WiFi module, and the analysis of the bearing condition can be initiated by using a web application. The SBC analyzes the received vibration signal automatically and publishes the feedback regarding the bearing condition to the Cloud. If the bearing has a defect, this will be captured in the Cloud-stored information, with information to characterize the nature of the defect.

CHAPTER 1. INTRODUCTION

In any continuous manufacturing process, maintenance is one of the most important non-value added, but necessary activities [1, 2]. The aim of today´s maintenance strategies is to predict the potential breakdown of a machine. Any unplanned downtime of a machine causes production loss and increased maintenance costs due to the sudden failure. A study of Deloitte illustrates that predictive maintenance increases the Equipment Effectiveness (EE) compared to proactive (75 %-90 %), planned (50 % - 75%) or reactive maintenance (< 50 %) up to more than 90 % of the original EE (OEE) (see Figure 1.1) [3].

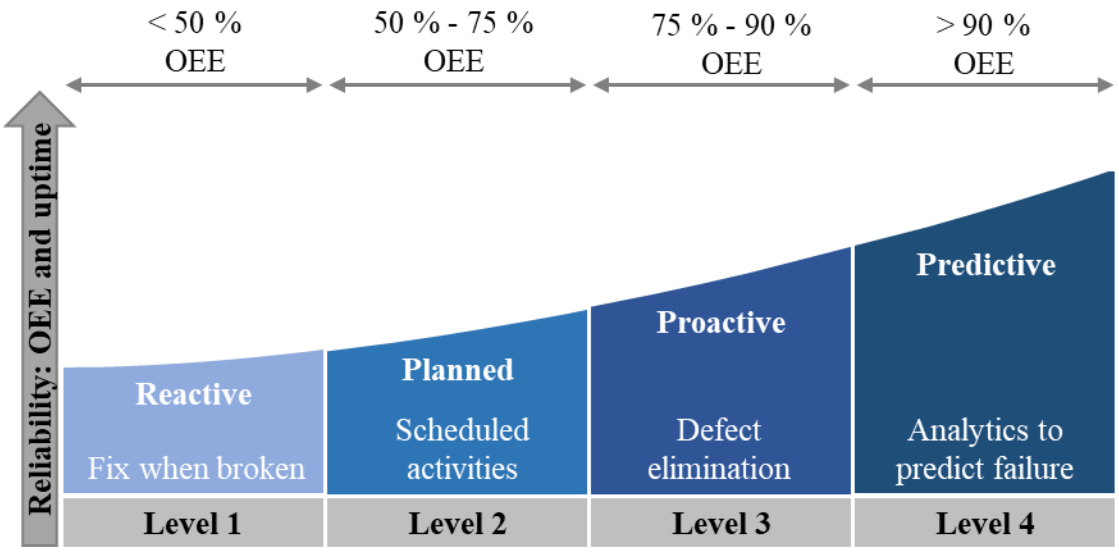

Figure 1.1: Effects of Maintenance Strategy on Equipment Effectiveness [3]

If an OEE of a machine is 100 %, this machine produces 100 % good parts, has a performance of 100 % and is 100 % available. If the OEE is lower, either the quality of parts, the performance or the availability are lower. This study shows that unplanned downtime of machines could cost industrial manufactures an estimated amount of

$ 50 billion each year [3]. The most modern and popular predictive maintenance technique is Condition-Based Maintenance (CBM). In CBM, specific machine parameters are measured to monitor the condition of a machine through its condition [4]. Since bearing failure is one of the most frequent causes of machine breakdown, it is necessary to measure the condition of any machine´s bearings. Rolling element bearings are among the most widely used components in machines, which is the reason this study focuses on this category of bearings [5–7].

In most instances, rotating machines have a unique vibration signature that relates to their health status. If a fault occurs in a bearing, the bearing changes its vibration signature and furthermore, it impacts the vibration signature of the machine during rotating operations [8]. This impact can be analyzed with the help of the measured vibration signal of the machine in the time or frequency domains.

1.1 Motivation

Presently, analyzing the vibration signal of a machine is intensive and requires experts to interpret the spectrum and the analysis results [8]. This is especially true for monitoring the bearing defects of many machines simultaneously. The extensive analysis required leads to a significant increase in cost and complexity due to the large amounts of vibration data that must be examined. Therefore, by automating the measured vibration signal analysis, the time and the cost can significantly be reduced. Especially when there is a large amount of vibration data, it is necessary to analyze these automatically in order to detect defects at an early stage. Additionally, the reliability of analyzing the bearing health can be improved by automating this process [9, 10].

1.2 Purpose

The purpose of this study is to reduce the complexity and cost of monitoring and analyzing the condition of rolling element bearings mounted in a machine. To realize this, a device was developed to measure the specific vibration signature of a machine. Furthermore, this device automatically analyzes the measured vibration signal and provides feedback as to the bearing condition. This device is portable and enables analysis of rolling element bearings in different machines. In addition, this device is internet enabled and is embedded into an Internet of Things (IoT) architecture. This IoT architecture can be used to execute the rolling element bearing analysis and to monitor its results in a database or in a web-based system.

1.3 Structure

In the next chapter, Chapter 2, the state-of-the-art of machine health monitoring and specifically measuring, analyzing and monitoring rolling element bearing defects are described. Moreover, the basic hardware components used for building the device and the software programs and algorithms used to analyze and monitor the rolling element bearing defects are explained.

The development of the device is examined in Chapter 3. It includes the decision process related to the hardware and software components. The assembly of the hardware components as well as the codes that are needed for the data acquisition and data analysis are shown.

Subsequently, an experimental setup for testing the developed device is explained in Chapter 4. The results and the factors influencing the data quality and data analysis are introduced.

In Chapter 5, the most important findings are summarized and the results are examined critically. This paper closes with an explanation of further research that is possible on the basis of the results of the experiments.

CHAPTER 2. STATE-OF-THE-ART

A device that measures the vibration signals of the rolling bearing elements and that analyzes these vibration signals automatically was developed. Furthermore, with the analysis results, the device shall provide a feedback as to the bearing condition to enable CBM. Therefore, it is necessary to distinguish CBM compared to the other maintenance strategies. After a brief description of rolling element bearings and their typical defects, the different ways of monitoring the rolling element bearing conditions and their use for CBM are explained. At this juncture, the methods of vibration signal processing are shown. Subsequently, the required hardware components and software tools for measuring and analyzing the vibration signal as well as for embedding the device into an IoT architecture are presented.

2.1 Maintenance Strategies

Maintenance strategies can be classified into Corrective Maintenance and Preventive Maintenance strategies [4, 11]. Preventive Maintenance can be divided into Time-Based Preventive Maintenance and CBM. Corrective Maintenance is the original maintenance strategy and is used to repair a machine after its failure. This strategy leads to a high production loss and high costs due to a sudden machine breakdown [4, 5]. If the maintenance is done regularly after a specific time interval, the maintenance strategy is called Time-Based Preventive Maintenance. Usually the time interval between the different repairs is shorter than the expected time between machine failures. Hence, the maintenance can be planned in advance and catastrophic breakdowns of the machine can be reduced. However, this maintenance strategy is not suitable for rolling element bearings, since their

breakdowns cannot be predicted with reasonable accuracy. Instead of that, there is a large statistical spread around the mean failure time of bearings[5]. Therefore, CBM is necessary to detect bearing defects at an early state and to enable a bearing to be replaced before it fails. CBM is a maintenance strategy with which the machine´s condition is monitored and analyzed. The analysis results illustrate defects and their development. Thus, maintenance activities can be performed just before a defect causes a failure and thereby, a possible catastrophic failure can be prevented [5, 12].

2.2 Rolling Element Bearing Defects

Rolling element bearings utilize the rolling action of rolling elements like balls or rollers to permit the rotation of a shaft relative to a fixed object [13]. These bearings have carrying capacities and low-friction characteristics. Therefore, rolling element bearings are frequently encountered in rotating machinery [12]. These bearings are used in production surroundings for the machine´s spindle. A sketch of the general structure of a rolling element bearing is illustrated in Figure 2.1.

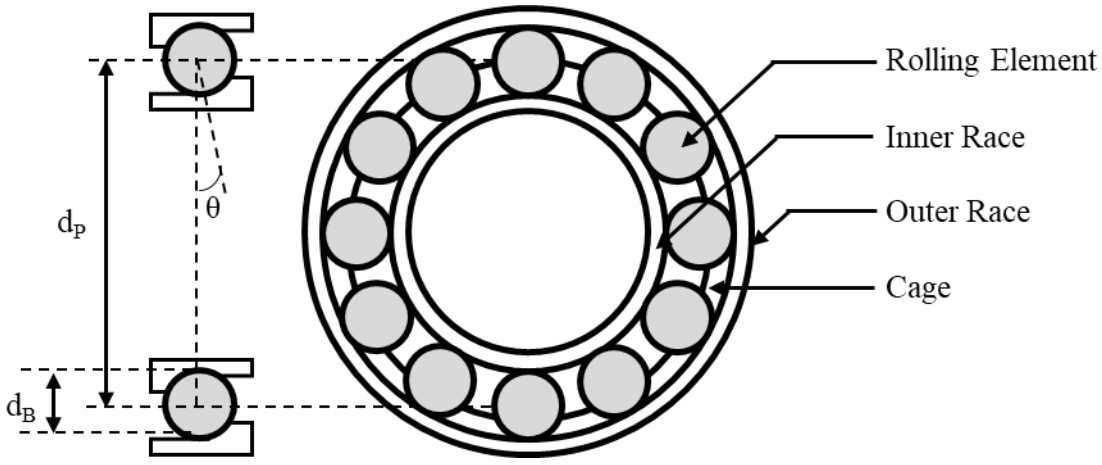

Figure 2.1: Structure and Characteristics of a Rolling Element Bearing

As illustrated in Figure 2.1, a rolling element bearing consists out of an outer race and an inner race between which the balls or rollers roll. The rolling elements are fixed on their location by a cage. d_P symbolizes the pitch diameter of the bearing, d_B the rolling element diameter and θ the contact angle between the rolling element and the inner race.

Rolling element defects can occur due to improper lubrication, improper mounting, misalignment, or if the bearing is not kept free of abrasives, moisture and corrosive reagents and if the bearing is overloaded. If all these causes are eliminated, defects of rolling element bearings can be caused by material fatigue. This material fatigue results from cracks that are below the surface. These cracks propagate to the surface and material particles are flaked off from the surface [14]. These worn off surfaces are a defect of the rolling element bearing. By reason of the structure of a rolling element bearing, such defects can occur on the outer race, the inner race, the rolling element or the cage [15]. An outer race defect is shown in Figure 2.2 a), a rolling element defect in Figure 2.2 b) and a cage defect in Figure 2.2 c). An inner race defect can look like an outer race defect, and this is why it is not illustrated in Figure 2.2. Rolling element bearings work under different conditions and often under heavy loading generated in the machinery; these differences are why these different defects occur [8, 12].

a) b) c)

Figure 2.2: Typical Rolling Element Bearing Defects [16, 17]

Bearing defects can be classified into distributed and local defects. Distributed defects include surface roughness, waviness, misaligned races and off-size rolling elements [18, 19]. These defects are caused by manufacturing errors, improper installation or abrasive wear [20]. Cracks, pits and spalls on the rolling surfaces are local defects [6, 21].

To detect such defects and to be able to predict the life of a rolling element bearing, CBM is necessary, since the life of a rolling element bearing is individual. If apparently identical rolling element bearings are exposed to the same conditions as, for instance, the same load, speed, lubrication, and environmental conditions, the individual bearings will not achieve the same fatigue life. By reason of this fatigue life dispersion, bearing manufactures indicate one or two properties of the same bearings to define their endurance [14]. These properties are

1. L_{10} which is the fatigue life that 90 % of the bearings will endure
2. L_{50} which is the median fatigue life that 50 % of the bearings will endure [14].

To be able to predict the fatigue life of an individual bearing, the vibration signal can be measured, since a local defect of a rolling element bearing causes a vibration signal. Each time the defect hits another element, an abrupt change in the contact stresses at the interface generates a pulse of a very short duration. This pulse produces vibration [21]. Moreover, there is a correlation between the location where the defect occurs (inner race, outer race, rolling element or cage) and its vibration frequency [5, 8, 21]. This correlation can be explained by means of Figure 2.3. This figure shows defects on the inner and the outer race as well as a defect on a rolling element at different times. At this juncture, t_1 is earlier than t_2 and t_2 is earlier than t_3.

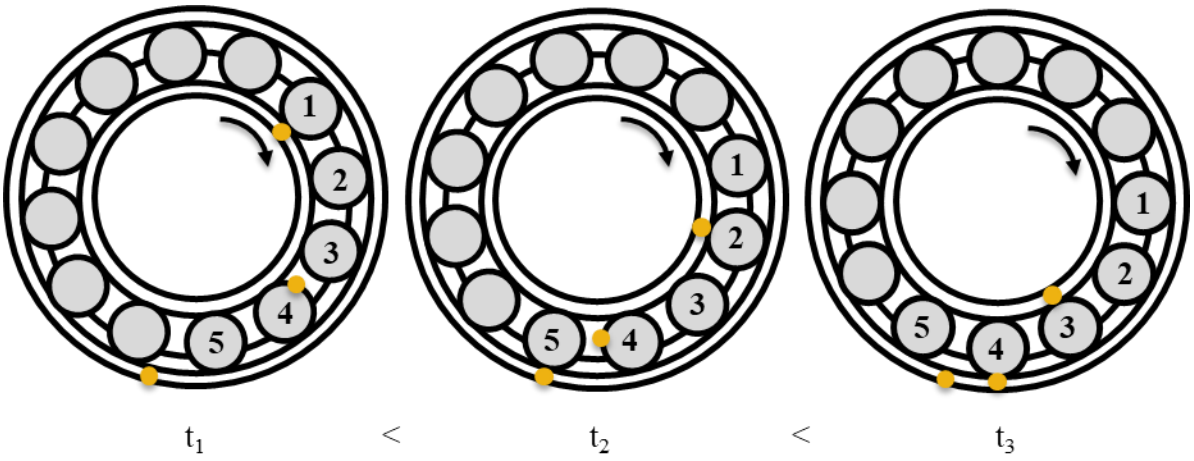

Figure 2.3: Correlation between a local Defect and a specific Vibration Frequency [5]

Due to the geometry of the bearing, the defect on the inner race hits another rolling element faster than the defects on the outer race or the rolling element hit another element. Between t_1 and t_3, the inner race defect hits three different rolling elements, whereas the defect on the rolling element only hits an inner or outer race two times. The defect on the outer race hits a rolling element just one time. Since these intervals between two hits differ related to the location of the defect, pulses are generated with different time intervals as well. Therefore, the vibration frequency has to be different. Moreover, it is intuitively reasonable that the time interval between two hits of a defect with another element depends on the spindle turning speed (respectively the spindle turning frequency). On the basis of these relationships, the characteristic frequency of a defect can be calculated. The formulas of the characteristic frequencies are shown in equation (1) - (4) [5, 10, 15]. The derivation of these four formulas will not be explained in this study, but further information may be found in the literature [22, 23].

$$BPFI = \frac{n}{2}\left(1+\frac{d_B}{d_P}\cdot\cos(\theta)\right)\cdot \omega_{Spindle} \quad (1)$$

$$BPFO = \frac{n}{2}\left(1-\frac{d_B}{d_P}\cdot\cos(\theta)\right)\cdot \omega_{Spindle} \quad (2)$$

$$BSF = \frac{d_P}{2d_B}\left(1-\left(\frac{d_B}{d_P}\right)^2\cdot\cos^2(\theta)\right)\cdot \omega_{Spindle} \quad (3)$$

$$FTF = \frac{1}{2}\left(1-\frac{d_B}{d_P}\cdot\cos(\theta)\right)\cdot \omega_{Spindle} \quad (4)$$

BPFI is the characteristic frequency of a defect on the inner race and is the abbreviation for ball pass frequency, inner race. Whereas, BPFO is the ball pass frequency, outer race. If there is a defect on the rolling element, the characteristic frequency is BSF which stands for ball spin frequency. A cage defect has the characteristic frequency FTF which is the abbreviation for fundamental train frequency. N symbolizes the number of rolling elements of the bearing and $\omega_{Spindle}$ is the rotational frequency of the machine´s spindle. The other symbols d_B, d_P, θ are explained in Figure 2.1. There is no slip considered in equations (1) - (4), this is why the measured characteristic frequencies of the defects usually deviate 1 - 2 % from the calculated values [5].

If the rolling element bearing has a defect, the vibration signal contains additionally the harmonics of these characteristic frequencies. The frequency of these harmonics is an

integer multiple of the characteristic frequencies [24]. Besides these characteristic frequencies and harmonics, the vibration signal can also contain sidebands around the harmonics [10]. These sidebands are generated by a modulation of the vibration signal which is caused by the local defect of an element of a rolling element bearing. The vibration signal is modulated by the rate at which the defect is passing through the load zone [5].

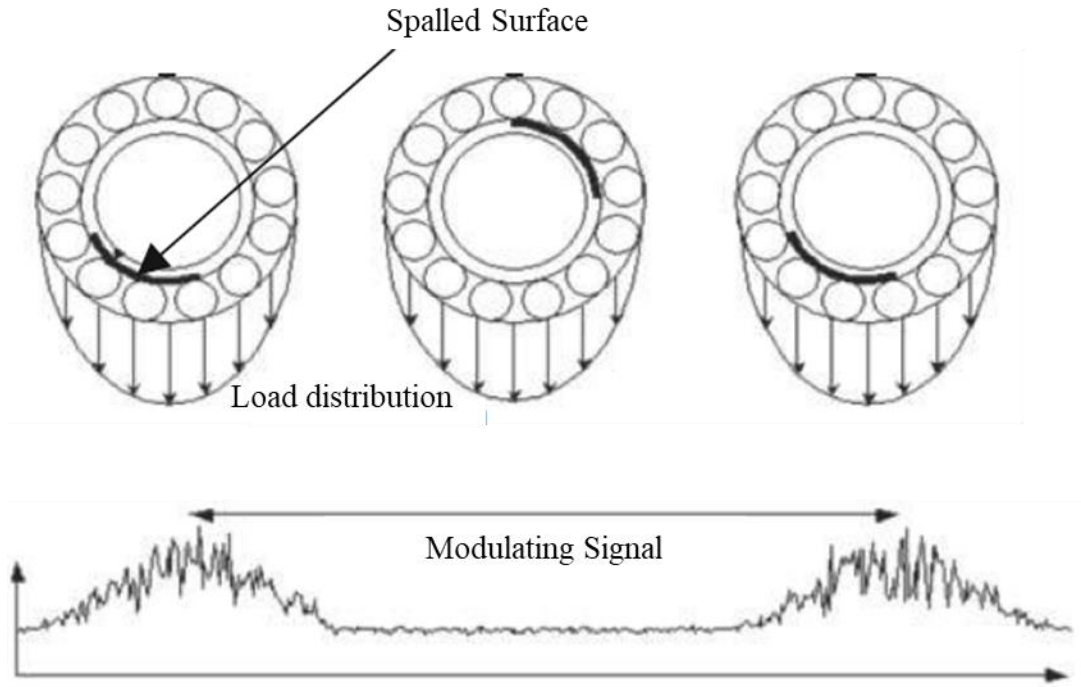

Figure 2.4: Modulation of the Vibration Signal [5]

Figure 2.4 illustrates the modulation of the vibration signal for a rolling element bearing with an unidirectional load [5]. If the outer race is static and the inner race rotates, the amplitude of the vibration shock that is caused by a defect of the outer race is always the same, since the load on this defect is always the same. If the inner race has a defect, the amplitude changes with the position of the defect due to the different loads at different positions. This correlation also applies for a defect of the rolling element. This defect changes its position as well. Since the cage does not bear any load, the amplitude of the

vibration shocks caused by these kinds of defect does not change. These different effects of the different types of defects are confirmed of Figure 2.5. This figure shows the vibration shocks caused by a defect of the rolling element bearing in the time and frequency domain.

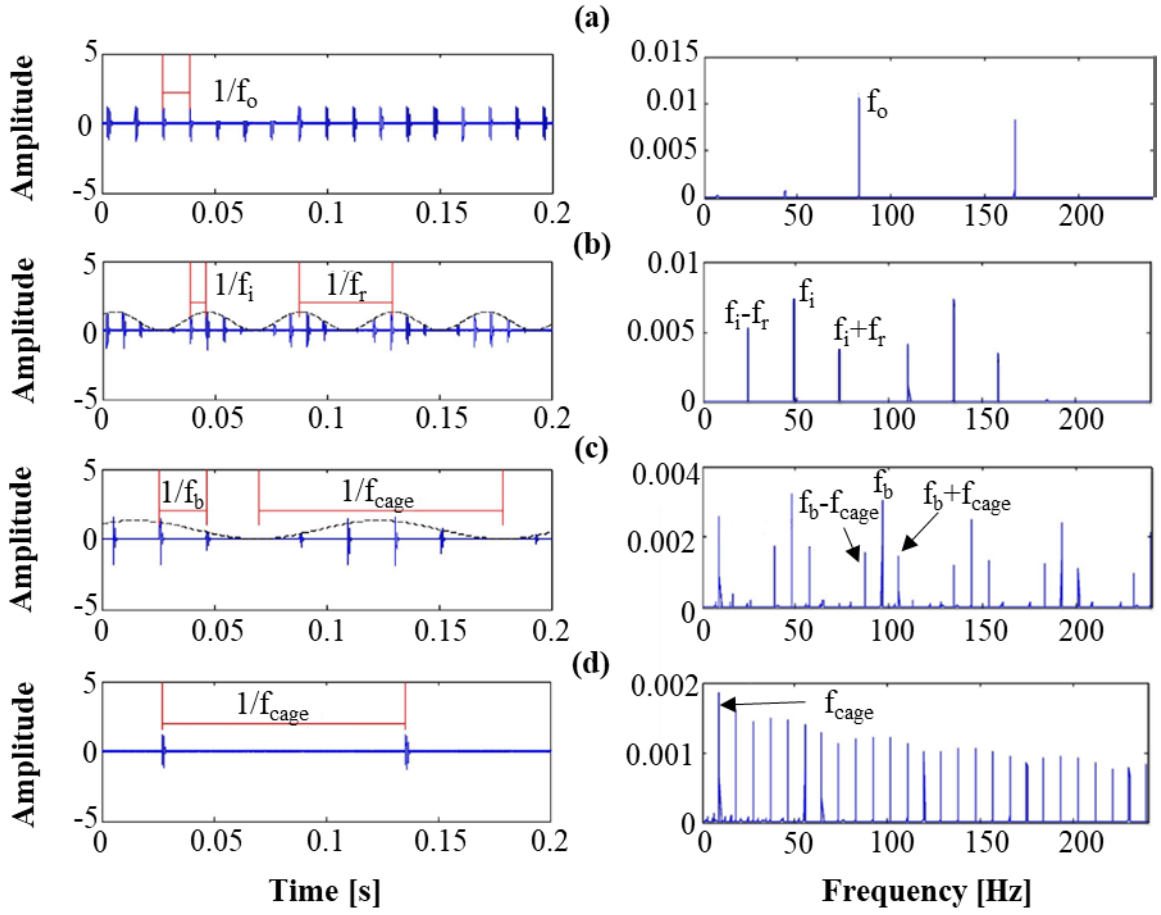

Figure 2.5: Modulation [15]

The amplitude of each frequency peak indicates the severity of the defect that influences the vibration signal. A higher amplitude implies a more severe defect [6].

In summary, the vibration signal generated from a rolling element bearing contains its structural dynamics and working conditions, since its defects become apparent and can

be classified. With the spectrum the details about a bearing fault as type, mode, growth and severity can be investigated [6].

2.3 Vibration Signal Processing

To investigate the condition of a rolling element bearing, the vibration signal influenced by this rolling element bearing has to be measured, processed and analyzed. Therefore, the different types of vibration transducers to measure the vibration signal as well as their advantages and drawbacks are explained. Moreover, the different methods to process and analyze the measured vibration signal are discussed.

2.3.1 Vibration Signal Measurement

Vibration can be expressed in the dimensions of displacement, velocity or acceleration. Due to these three dimensions there exist displacement, velocity and acceleration transducers with which the vibration signal can be measured [5, 6, 25]. The frequency range, accuracy and the location of the transducer on the machine has to be considered for choosing an appropriate transducer for analyzing the health of rolling element bearings [26].

Displacement can be measured with proximity probes or with displacement laser sensors [5, 27]. If the displacement is measured by proximity probes, two proximity probes are installed in the bearing cap and due to the vibration these two probes change their relative positions. Thereby, the relative displacement between the probe tips is measured. These measurements can be based on capacitive, magnetic or electric properties [5]. Although, proximity probes are capable to measure vibration frequencies of more than

2 kHz, the amplitudes of the vibrations above 1 kHz are extremely small. Therefore, these amplitudes can usually not be distinguished from noise [28]. For investigating defect frequencies of rolling element bearings and their harmonics, frequencies above 1 kHz have to be detected, though. Hence, proximity probes cannot be used for analyzing rolling element bearing defects. A higher frequency range can be achieved by using displacement laser sensors, since they have a large measurement range. Moreover, these sensors have a high accuracy and a good stability. The drawbacks of displacement laser sensors are on the one hand that a calibration of the laser beam is necessary to ensure its high accuracy and on the other hand that such a sensor has a very high price [29, 30]. Since the purpose of this Thesis is to use the bearing health monitoring system for a large number of machines, a high price of one sensor accumulates if many of these systems are built. Therefore, a less expensive sensor should be used.

The natural frequency of velocity transducers is the limit of the frequencies that can be picked up with these kind of transducers. Thus, the frequency range of velocity transducers is usually between 10 and 2000 Hz [5, 28]. Moreover, the vibration measurements of velocity transducers are directional, because the measured signal is dependent on the position and direction of this transducer [28]. For analyzing rolling element bearing defects, a higher frequency limit as well as measurement results regardless of the direction of the transducer are needed. The measurements have to be independent of the direction to ensure the same results for different measurements on different machines. Therefore, velocity transducers cannot be used for analyzing rolling element bearing defects.

Accelerometers are mounted on the casing of the part from which the vibration signal is picked up, and these transducers produce a signal proportional to the measured acceleration [5, 28]. The advantages of using an accelerometer are that it has high accuracy, light weight, good temperature resistance, that it emphasizes high frequencies and most important that it has a wide frequency range up to 10 kHz [5, 27, 28].

Table 1: Characteristics of different Transducer Types

Characteristics	Displacement Transducer	Velocity Transducer	Acceleration Transducer
Frequency range	< 1 kHz (Probe) > 2 kHz (Laser)	< 2 kHz	< 10 kHz
Emphasizing of signal	Emphasizes low frequencies	-	Emphasizes high frequencies
Installation	Extensive (Laser)	simple	simple
Measurement Quality	High accuracy (Laser)	Only directional measurement	High accuracy

Due to the high accuracy, the wide frequency range and the amplification of high frequencies, this transducer is appropriate for measuring vibration signals that are used for analyzing rolling element bearing defects. However, the drawbacks of the accelerometer are that it has to be mounted on the bearing housing to be able to pick up the vibration signal caused from these bearings. Moreover, it is highly sensitive to noise and easily influenced by ambient noise under actual factory running conditions [27]. There exist different types of accelerometers like piezoelectric, piezoresistive, capacitive and micro electro-mechanical systems (MEMS) accelerometer. MEMS accelerometers are appropriate for rolling element bearing analysis, since these kinds of accelerometers have frequency ranges of up to 10 kHz and provide a low cost installation due to their simple

set-up [31–34]. MEMS accelerometers are based on a mass-spring system and are often analog devices [34]. When exposed to ground acceleration, the casing moves relative to an inertial mass. This displacement is measured by a companion application-specific integrated-circuit (ASIC) chip which includes a capacitive position sensor. This sensor measures the capacitance between the mobile electrodes attached to the inertial mass and those attached to the frame. The MEMS accelerometer generates an output voltage that is relative to this capacitance. By sampling this output voltage, the acceleration can be calculated [35].

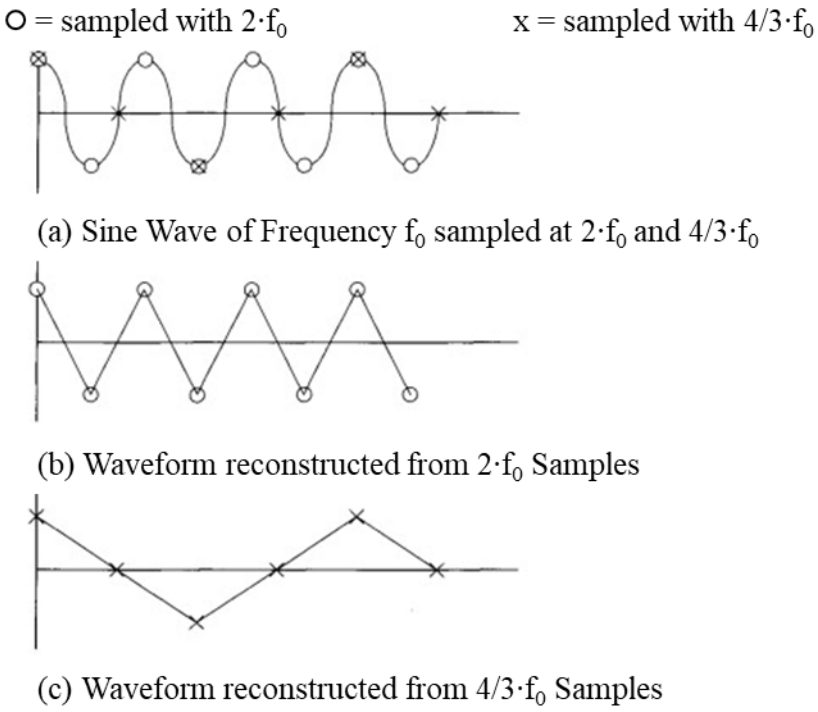

Figure 2.6: Example of Aliasing [36]

If the output voltage is analog and gets sampled, it has to be digitized by an analog-to-digital converter (ADC). To be digitized accurately by the ADC, the signal has to be sampled at least two times higher than the highest frequency component in the signal.

If the signal is sampled at a lower frequency than two times the highest frequency component, the digitized waveform is distorted. This distortion is called aliasing [36, 37]. This aliasing can be shown by Figure 2.6.

This figure shows a signal in the time domain that consists out of one frequency component f_0. If the signal is sampled with a sampling frequency that is equal to the signal frequency f_0, the digitized waveform would be a constant value. If the signal is sampled with a higher sampling frequency, but still lower than two times f_0, for instance $4/3 \cdot f_0$, the digitized waveform has a frequency of $1/3 \cdot f_0$. This result of aliasing is the difference frequency between the sampling rate and the signal frequency, which is $(4/3 - 1) \cdot f_0$. If this signal is sampled at two times f_0, the sample would produce a waveform with exactly this frequency. In this case, the only distortion is that the digitized waveform appears to be a triangle wave instead of a sine wave, as in Figure 2.6 b). If the signal is sampled with a higher frequency than two times f_0, the digitized waveform would converge to the analog signal [36]. This is the reason why traditionally the factor of the sampling rate is 2.56 times higher than the highest frequency component [37]. Summarized, just frequency components with a frequency not higher than half of the sampling frequency f_s can be digitized properly. This frequency $f_N = 0.5 \cdot f_s$ is called Nyquist frequency, since this sampling theorem was presented by Nyquist in 1928 [36, 37].

This sampling theorem leads to the requirement that the sampled signal does not contain frequencies that are higher than the Nyquist frequency f_N. Otherwise, the result will be a distorted digitized signal. Therefore, either an analog antialias filter has to be used or the sampling frequency f_s has to be more than two times higher than the bandwidth of the analog sensor [37]. For instance, if an analog sensor has a bandwidth of 2 kHz, then the

sampling frequency f_s has to be higher than 4 kHz to avoid aliasing. If the sensor´s signal is sampled with a frequency of 3 kHz, an antialias filter with a cutoff frequency below $0.5 \cdot f_s$ has to be used to avoid aliasing. The slope of any analog filter above the cutoff frequency is finite. This ratio between the sampling frequency and the cutoff frequency of the antialias filter is called the oversampling ratio [37].

2.3.2 Vibration Signal Processing and Analyzing

After measuring the vibration signal, the signal has to be analyzed to assess the condition of the rolling element bearings. There exist different analyzing techniques in the time and frequency domain with which different aspects of the rolling element bearings and the machine´s spindle can be investigated [38, 39]. An overview over the analyzing techniques discussed in this chapter is shown in Figure 2.7.

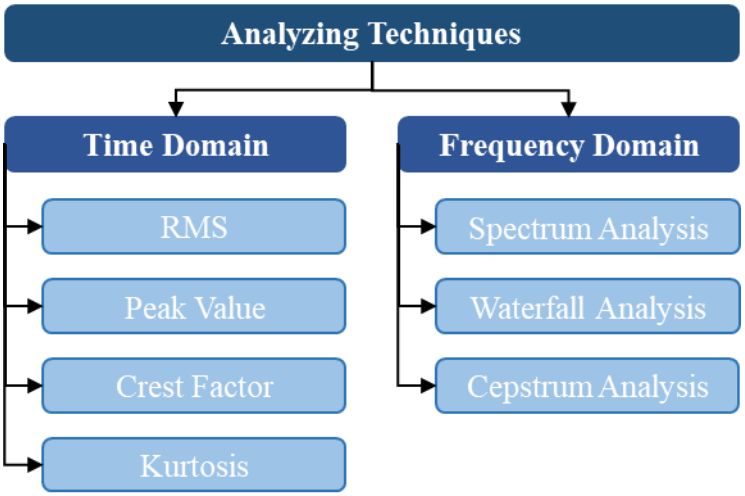

Figure 2.7: Overview over different Analyzing Techniques

Root-Mean-Square (RMS), Peak Value, Crest Factor and Kurtosis analysis are analyzing techniques of a rolling element bearing in the time domain. Analyzing techniques

in the frequency domain are the spectrum analysis, the waterfall analysis and the cepstrum analysis.

The RMS measures the energy level of the vibration signal and can be calculated with equation (5) [40].

$$RMS = \sqrt{\frac{\sum x_i^2}{N}} \tag{5}$$

N is the total number of samples and x_i represents the vibration signal of each sampled point. The value of the RMS changes relative to the working conditions and the appearance of rolling element bearing defects [40, 41]. Moreover, with this technique it is possible to distinguish different types of bearing defects. For instance, Tandon has shown that the overall RMS level of a defect on the outer race is higher than of a defect on the inner race [42]. The advantage of RMS is, that it is simple to calculate and to use for historical analysis as the overall RMS level is plotted over time. The drawback is that the RMS level of a healthy machine has to be known to detect differences. Moreover, not every rolling element bearing defect can be detected. Tandon shows exemplarily that only outer race defects with a diameter larger than 75 µm and inner race defects with a diameter larger than 200 µm have been detected [42].

Another analysis technique in the time domain is the peak value that is calculated with equation (6):

$$P_v = \frac{1}{2} \cdot [\max(x_i) - \min(x_i)] \tag{6}$$

The peak value of the measured vibration signal is calculated by the subtraction of the sampled point with the minimum amplitude of the sampled point with the maximum amplitude. This value is then multiplied by two [9]. Tandon has shown that the peak value increases as the diameter of a bearing defect increases. However, the drawback is that only rolling element bearing defects with a diameter larger than 100 μm (outer race defect) or larger than 200 μm (inner race defect) can be detected [42]:

Another time domain analysis technique is to calculate the crest factor and to monitor this factor over time to investigate its trend. The crest factor can be calculated by dividing the peak value P_v by the RMS [9, 43]:

$$\text{Crest Factor} = \frac{P_v}{\text{RMS}} \tag{7}$$

The value of the crest factor increases as the amplitude of high frequency impacts in the bearing increase compared to the amplitude of broadband overall vibration. The advantage of the crest factor is that it does not vary much at different speeds. Therefore, it can be used if the machine runs often with different speeds [44]. However, the effectiveness of the crest factor decreases as the spindle turning speed is higher than 3000 rpm. Therefore, it can only be used for speeds that are lower than 3000 rpm [45]. Moreover, different authors showed that not all the defects can be detected with the crest factor [12, 21]. For instance Tandon could detect an inner race defect but not an outer race defect with this technique [42].

Kurtosis is an indicator for rolling element bearing defects, too. It can be calculated with equation (8) [9, 12]:

$$\text{Kurtosis} = \frac{\sum(x_i - \text{mean}(x))^4}{N \cdot \text{RMS}^4} \tag{8}$$

For a new bearing, this Kurtosis value should be close to three. If a defect occurs, it generates impulses that change the distribution of the vibration signal and that therefore lead to a higher Kurtosis value. Once the damage gets greater than the spacing of the rolling elements, the continuous shock load would cause a normal distribution of the signal and therefore, a reduction of the Kurtosis value to the value three [41]. In general, the Kurtosis value can be a good indicator for bearing defects at low spindle turning speeds. If the spindle turning speed increases, the Kurtosis value loses its ability to indicate a bearing defect [12].

Besides analyzing the vibration signal in the time domain, analysis techniques in the frequency domain can be applied. In favor, the vibration signal has to be transferred from the time domain to the frequency domain. This transformation can be achieved by using the fast Fourier transform (FFT) [46–48].

The FFT is a method for efficiently and quickly computing the discrete Fourier transform (DFT), which maps a sequence of length L in the time domain to its spectrum of length L in the frequency domain [47, 48]. Thereby, a vibration signal in the time domain is decomposed and the different frequencies which overlay in the time domain are visible in the frequency domain. This representation of the frequency components in the frequency domain is called the spectrum. The result of an FFT is illustrated in Figure 2.8. The left graph shows a signal with a frequency of 10 Hz, and this can be directly seen with the right graph, since there is only one peak at exactly 10 Hz. The right graph is the result of the FFT of the left graph.

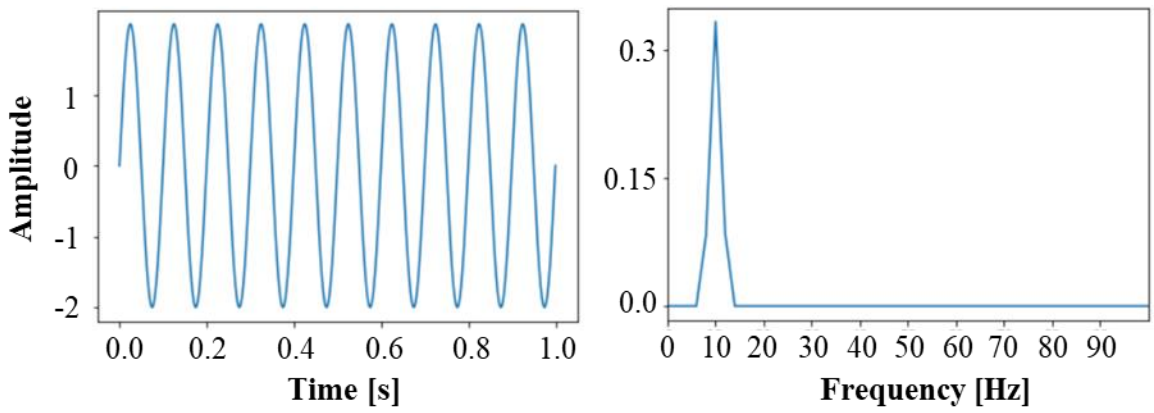

Figure 2.8: Illustration of a Signal in the Time- and Frequency-Domain

The FFT is not derived in this Thesis. However, this work is based upon mathematical properties that are entirely analogous to those of the Fourier integral transform [47].

There exist three typical pitfalls when an FFT is performed: the picket fence effect, aliasing and leakage. The continuous spectrum is discretely sampled in the frequency domain and it is not necessarily sampled at its peaks, hence the picket fence effect occurs. It is as the spectrum is viewed through the slits in a picket fence [5, 46].

The phenomenon of aliasing is already discussed in Chapter 2.3.1. However, the effects of aliasing on the representation of the vibration signal frequency domain will be pointed out. If the sampling frequency f_s is not high enough, high frequency components of the vibration signal can impersonate low frequencies and therefore, incorrect frequency components are displayed in the frequency domain. Hence, the sampling frequency has to be chosen high enough to ensure that the frequency components are displayed correctly in the spectrum [5, 46].

The problem of leakage is inherent in the FFT of any finite record of data, since the FFT is applied on a certain time interval Δt between t_1 and t_2. The record of data before t_1 and after t_2 is neglected. The effect of leakage on the spectrum is shown in Figure 2.9.

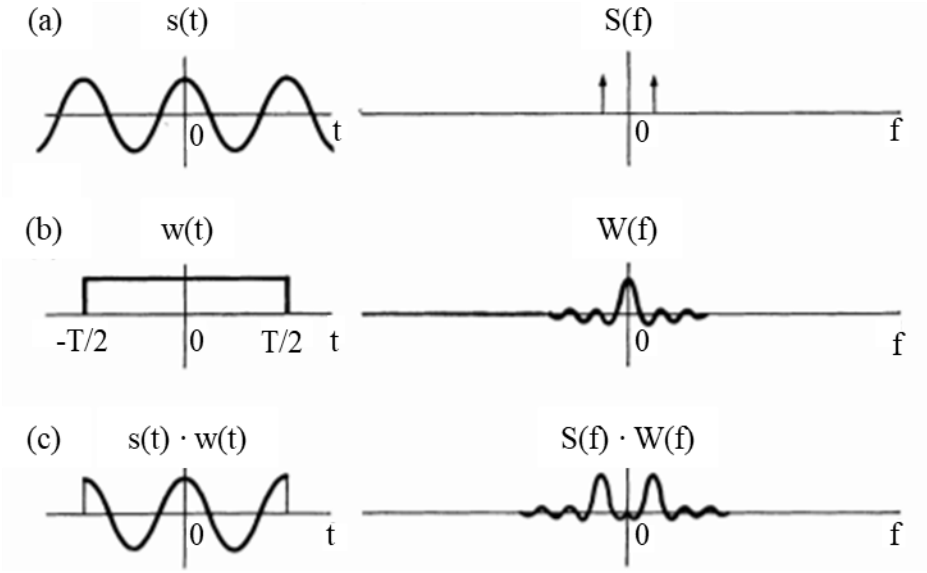

Figure 2.9: Effects of Leakage on the Results of the FFT [46]

Figure 2.9 a) shows a signal s(t) in the time domain, whereas Figure 2.9 b) shows the signal in the frequency domain. The signal in b) is ideal, though, since the FFT needs a data window with finite data records on which it is applied. This window is represented by w(t) in c). The representation of w(t) in the frequency domain is illustrated by W(f) in d). The result of applying the FFT on s(t) by using the window w(t) is depicted in f). It can be seen that the window function w(t) influences the result and generates a series of spurious peaks that are called side lobes [5, 46].

There exist various kinds of windows as for instance the rectangular window that is shown in Figure 2.9, the Hanning window, the Hamming window, the Blackman window, the Blackman-Harris window, the Rife-Vincent window and the Nuttall window [49, 50].

Only the Hanning window is explained in this Thesis, since it has the highest side lobe roll-off rate. This is important for vibration signals [51]. Moreover, the Hanning window reduces the effects of leakage and has a good frequency resolution, because it generates a better signal representation in the frequency domain. This is investigated by Cerna and Harvey in [51]. The Hanning window function is defined as

$$w_h(t) = 1 - \cos(2\pi t/T) \tag{9}$$

where $T = N \cdot \Delta t$ is the sampling duration. N is the number of samples and Δt the duration of each sample [52]. The Hanning window is illustrated in Figure 2.10.

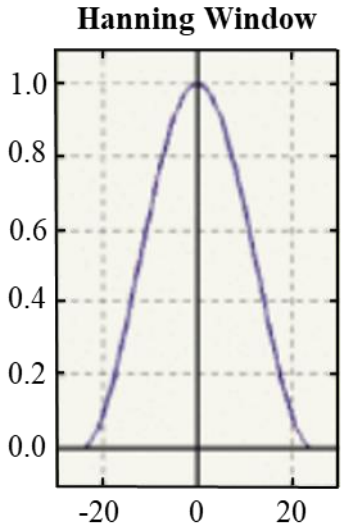

Figure 2.10: Hanning Window [53]

The conversion between the time and the frequency domain involves some compression of data, and it has the advantage that the data can be averaged easily and thus the effects of random noise can be reduced. Therefore, a data point averaging is useful

before the actual FFT is undertaken. A drawback of applying the FFT is that there is a loss of some amount of information from the vibration signal [38, 54].

Already analyzing the frequency spectra generated by the FFT is a useful analysis technique for condition health monitoring of rolling element bearings. The spectrum consists of characteristic vibration frequencies, such as the spindle turning speed or defect frequencies, if a defect appears. Moreover, the harmonics of these characteristic frequencies can be detected early in the spectrum, can be diagnosed accurately and trended over time [54, 55]. Thus, there is a higher chance to detect changes of frequencies with monitoring of frequency spectra, rather than overall levels as for instance RMS. Significant changes in individual frequency components only affect the overall vibration levels at the very last stages [5].

These spectrums can also be used for a waterfall (or cascade) analysis. As discussed in Chapter 2.2 the frequencies of rolling element bearing defects depend on the spindle turning speed. The waterfall analysis utilizes this relationship. Different spectrums that show vibration signals of different measurements with different spindle turning speeds are plotted in 3D. The x-axis shows the frequencies, the y-axis the different spindle turning speeds and the z-axis the amplitude. Hence, the peaks of the defect frequencies change along the y axis, and it can be guaranteed that the defect exists [56, 57].

Another technique to analyze the vibration signal of a rolling element bearing is the cepstrum analysis. The cepstrum is generated by applying the FFT to the logarithmic converted spectrum of a vibration signal. This analysis technique is mainly used for analyzing signals that contain families of harmonics or sidebands. In this connection, it is

important that the harmonics or sidebands have uniform spacing. Otherwise, the cepstrum analysis does not work properly. If there exist harmonics in a spectrum, these harmonics are represented in the cepstrum by one peak. This analysis technique has its advantages if the harmonics or sidebands of a signal are not well distinguishable from other peaks in the spectrum. In the cepstrum only these harmonics or sidebands are displayed and all other peaks that are shown in the spectrum are not represented in the cepstrum. The drawback of the cepstrum is that defect harmonics cannot be detected if these harmonics are completely immersed in noise in the spectrum. Moreover, the cepstrum analysis requires sidebands and harmonics with a good resolution in the spectrum [5].

In general, the analysis techniques in the time domain monitor parameters that focus on the overall energy level of the vibration signal that is generated by the rolling element bearings. These are used to trend the overall level over time and if this increases a defect is indicated. Thus, these analysis techniques are simple to trend and can be used to obtain general information about the bearing health. However, if the rolling element bearing is analyzed in detail and the defect type must be indicated, analysis techniques in the frequency domain have to be applied. By using the frequency domain representation of a vibration signal, the different frequency components of the signal can be distinguished. Therefore, a detailed analysis of the rolling element bearing is possible. The simplest form of analyzing the vibration signal in the frequency domain is to analyze the spectrum and the contained frequency peaks. However, if the frequency peaks are difficult to distinguish, another analysis technique as the cepstrum or waterfall analysis can be applied.

2.4 Components for the Device

A device is developed that measures and analyzes the vibration signal automatically and that provides feedback as to the rolling element bearing condition. Thus, this device detects defects in the bearing and moreover, it is embedded into an IoT architecture. Necessary hardware components, such as sensors, microcontroller and single-board computer (SBC), are exemplified to achieve this purpose. The communication protocol Universal Asynchronous Receiver-Transmitter (UART) and Message Queuing Telemetry Transport (MQTT) are illustrated. With these communication protocols different hardware devices can communicate, and it becomes feasible to embed the device in an IoT architecture. Moreover, programming languages that are needed to process and analyze the vibration signal are explained. In detail, the programming languages Python and C are presented.

2.4.1 Hardware Components

At first, the device must be able to detect and to measure the vibration signal, so a sensor is needed. A sensor is a device that measures environmental conditions such as, for instance temperature, vibration, acoustic or electrical signals. A classification and an overview about the different types of sensors is given in [58]. Usually, a sensor converts the measured environmental condition into an analog or digital signal with which the environmental condition can be interpreted by devices such as microcontrollers or SBC. These signals can be transmitted with an analog signal or with communication protocols, such as for instance UART.

A microcontroller, also called microcontroller unit (MCU) is a small computer on a single integrated circuit. It contains processor cores (CPU), memory storage and programmable input/output peripherals. These devices often have program memory, as for instance random-access memory (RAM), and use popular communication protocols, as for instance UART, to establish a link between the MCU and other hardware devices. MCUs have a real time clock (RTC) and their system is deterministic, so a sensor can be sampled with precise time steps. An MCU can be programmed in assembly language or high-level programming languages as C, Python or JavaScript [59].

An SBC uses a microprocessor as the primary computational engine and consists of local memory, input and output devices and often an interface to a common bussing scheme [60]. A typical application of an SBC is in the industrial context, since these devices are smaller than motherboards that are used in computers. The differences between an MCU and an SBC are that the SBC has more memory and storage, it can run an operating system completely and therefore, these devices are more powerful.

2.4.2 Software Components

The measured vibration signal must be transmitted from the sensor to another hardware device, so the serial communication protocol UART is introduced. Moreover, the signal must be processed and analyzed. For these, the programming languages Python and C are explained. To embed the device into an IoT architecture, publish/subscribe protocols, especially MQTT and the programming tool Node-Red are illustrated.

UART is a serial communication protocol that enables communication between different devices. UART is an asynchronous communication protocol, so the data is only

sent as needed. Thus, there is no continuous data stream. With the UART serial transmission, one bit is sent at a time. This is illustrated with Figure 2.11. In this figure a transistor-transistor logic (TTL) is shown. This logic is based upon a 3.3 V (or 5 V) principle. The signal is high if the voltage has the value 3.3 (or 5 V). If the voltage has the value 0 V, the signal is low. Before the actual signal is transmitted, a start bit is sent. After the actual signal, a stop bit is sent, so the end of the current frame is signified.

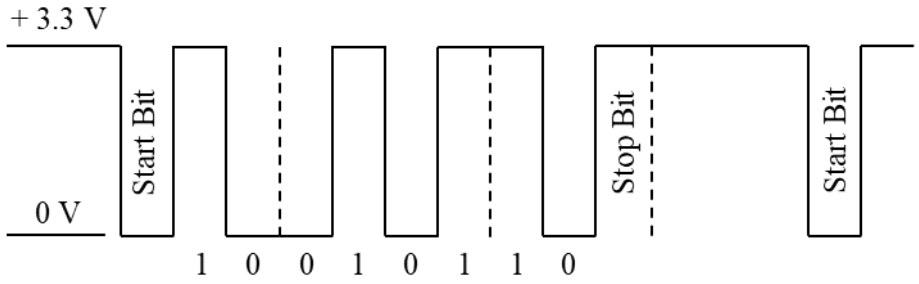

Figure 2.11: TTL with Start and Stop Bit

Python is a high-level, general purpose programming language that was created by Guido van Rossum and first released in 1991. The main purpose of Python was to develop a readable code and to have syntax with which the program can be programmed with fewer lines. Moreover, the interpreters of Python are open-sourced and currently, Python is widely used for many different applications as for instance data analysis [61].

C is a general purpose, imperative programming language. This programming language provides constructs that map efficiently to typical machine instructions, so it is often used to program embedded devices such as MCUs. C is not object orientated, so for instance on SBC other object orientated programming languages such as Python or C++ are preferred.

Publish/subscribe protocols are asynchronous service-to-service communication protocols used in serverless architectures. A serverless architecture is a cloud-computing execution model in which the cloud provider runs the server. Any message that is published to an event service with a specific topic is through this event service immediately received by all subscribers of this topic. This is called a topic-based publish-subscribe architecture and is an event-based interaction with the advantage of decoupling of space, time and synchronization between publisher and subscriber. Space decoupling means that the publisher and subscribers do not need to know each other. Usually, neither the publisher nor the subscriber holds references of the other, and the publisher also does not know how many subscribers exist. The time is decoupled, since the subscriber does not need to be connected while the publisher publishes a message. The subscriber can receive the notification of the message after being reconnected, and during this time the publisher can be disconnected. Publish/subscribe protocols are decoupled of synchronization, since the publisher is not blocked while publishing messages, and the subscriber can receive the messages asynchronously [62, 63].

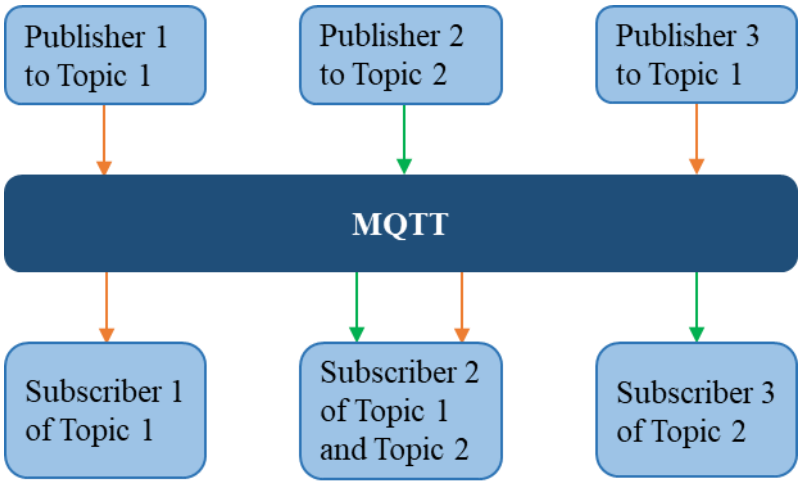

Figure 2.12: Figure of Topic-Based Publish-Subscribe Architecture with MQTT

MQTT is such a publish/subscribe protocol and uses the topic-based publish-subscribe architecture that can be seen in Figure 2.12. This figure illustrates three publishers that publish messages either to topic 1 (orange arrow) or to topic 2 (green arrow) and three subscribers that subscribe from topic 1 (orange arrow) or from topic 2 (green arrow). The reliability of messages in MQTT is based upon three Quality of Service (QoS) levels. QoS level 0 means that a message is delivered at once, and no confirmation of the reception is required, whereas in QoS level 1 such a confirmation is required. In QoS level 2, a four-way handshake mechanism is used for the delivery of messages exactly once [64].

Node-Red is a programming tool with which different hardware devices, application programming interfaces (APIs) and online services can be wired together using a web-based editor [65]. Moreover, it is a visual programming tool for IoT and therefore, deep programming knowledge is not needed. An exemplary Node-Red flow is illustrated in Figure 2.13.

Figure 2.13: Exemplary Node-Red Flow

Node-Red is event driven, so an event starts a flow. In Figure 2.13, this event is initiated by the blue node, which is called the inject node. It inputs two hexadecimal numbers (a32 and b4c) into the flow. The next node, the split node, splits these numbers into two different numbers and puts these into the function node. This function node converts these hexadecimal numbers into integers. These function nodes are programmable

and can be programmed with JavaScript. Subsequently, the purple node publishes the integer numbers to the cloud and to the database using MQTT.

Node-Red has the advantage that it is open-sourced and that it fits on the most widely used SBCs, as for instance the Raspberry Pi, the BeagleBone Black and the Arduino. Moreover it can connect different MCUs, as for instance the Particle Photon using the Particle Cloud or other MCUs, using a UART connection. If Node-Red is installed on an SBC, it can be open web-based from any computer, just the IP-address of the SBC and the port of Node-Red have to be known. Related to measuring and analyzing the rolling element bearing condition, it has the advantage that the condition can be monitored location-independently.

The disadvantages of Node-Red are that it is asynchronous. This means that the data are not transmitted in regular intervals, and the response time is unpredictable. Besides this, it has a slower performance than for instance Python, C++ or Java and therefore, no real-time performance.

However, Node-Red is a web-based programming tool for IoT applications and provides a lot of possibilities, if there is no need for high performance. As for instance for monitoring the rolling element bearing condition it is a useful tool, since it can be programmed easily, can be controlled location-independent and it can wire the different hardware devices and programming tools together.

MCUs have a deterministic system and suit very well for a precise sampling of sensors, whereas they are limited by the programs they can run. SBCs can run different programs and can be programmed in different languages. Therefore, they can be embedded

into an IoT architecture and are more powerful to analyze data. For the communication between an MCU and an SBC, UART can be used, why this communication protocol is explained. Python is a programming language with which data can be analyzed efficiently on SBCs. MCUs have to be programmed in the programming language C, why this language is presented, too. Node-Red is illustrated, because it can be handled easily, can tie different hardware and software components together and therefore, it can embed the system into an IoT architecture. With MQTT, the results can be published to the cloud and hence, stored in a database. Moreover, MQTT is integrated in Node-Red, why it is used.

CHAPTER 3. PROPOSED FRAMEWORK OF THE VIBRATION MEASUREMENT BOX

This chapter introduces the proposed vibration measurement box with which the vibration signal can be measured and analyzed, and which provides feedback as to the bearing condition. At first, the proposed hardware components, and subsequently the software used for the data acquisition and data analysis are presented.

3.1 Proposed Hardware Components

The used hardware components include the specific sensor, the MCU, the SBC and the assembled measurement box. Moreover, the reasons for choosing the specific component are explained.

3.1.1 Sensor ADXL203EB

For measuring the vibration signal of rolling element bearings properly, frequency ranges higher than 2 kHz are preferred. Neither the displacement nor the velocity transducer can measure frequencies higher than 2 kHz. Moreover, accelerometers have a high accuracy and emphasize high frequencies. This is important, since most of the rolling element bearing defect frequencies are in the higher frequency range. Therefore, an accelerometer is chosen (cf. Chapter 2.3.1).

Different properties of accelerometers must be considered in choosing a proper accelerometer. The sensitivity, the bandwidth (also called frequency range) and the spectral noise influence the measured signal and its quality. The number of measured axes is

important for measuring more vibration data and to be sure that both axes work properly and output the same vibration signal. Moreover, the price must be considered, since with this type of measurement box, the rolling element bearing conditions of many different machines will be monitored. Therefore, a less expensive accelerometer significantly reduces the costs.

ADXL203EB ADXL1001Z

Figure 3.1: Direction of the Measured Axis of the Accelerometer [66, 67]

Two different accelerometers, the ADXL203EB and the ADXL1001Z, shown in Figure 3.1, are compared to assess the sensitivity that is needed for proper vibration measurements. The characteristics of these accelerometers are shown in Table 2 and are copied from the accelerometers´ datasheets [66, 67].

Table 2: Characteristics of the ADXL203EB and the ADXL1001Z

Characteristics	ADXL203EB	ADXL1001Z
Sensitivity	1000 mV/g	20 mV/g
Bandwidth	0.5 – 2,500 Hz	11,000 Hz
Spectral Noise	110 µg/(Hz)$^{0.5}$	30 µg/(Hz)0.5
# Measured Axis	2	1
Price	$ 36.26	$ 75.00

Both accelerometers are produced by Analog Devices. The first one is the ADXL203EB and has a sensitivity of 1000 mV/g. The second accelerometer is the ADXL1001Z and has a sensitivity of 20 mV/g.

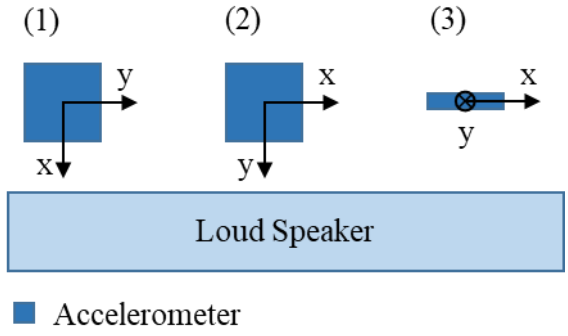

Figure 3.2: Experimental Set-Up to investigate the Sensitivity of each Accelerometer

As shown in Figure 3.2, the accelerometers are mounted in different positions (1, 2 and 3) on a loud speaker and used to measure the vibration of the loud speaker. The loud speaker plays a signal with a frequency of 2 kHz. The measured vibration signal is then transformed from the time domain to the frequency domain by using the FFT. The particular signals in the frequency domain are illustrated in Figure 3.3 and Figure 3.4.

In Figure 3.3, the highest amplitude (more than $4 \cdot 10^{-7}$) is detected at 2 kHz. This shows that the measured signal of accelerometer ADXL203EB detects the frequency of 2 kHz that is input into the loud speaker. The other frequency components are noise, which is for instance the high peak (amplitude of more than $3 \cdot 10^{-7}$) at a frequency of 60 Hz. This frequency peak at 60 Hz is caused by electrical noise.

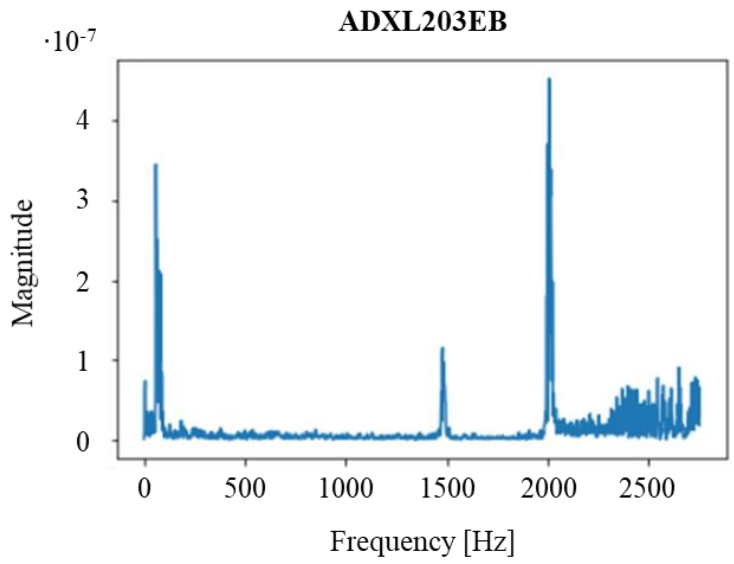

Figure 3.3: Measured Vibration Signal of the ADXL203EB in the Frequency Domain

On the other hand, in Figure 3.4 there are several peaks with different amplitudes. The frequency of the vibration signal measured by the ADXL1001Z cannot be detected.

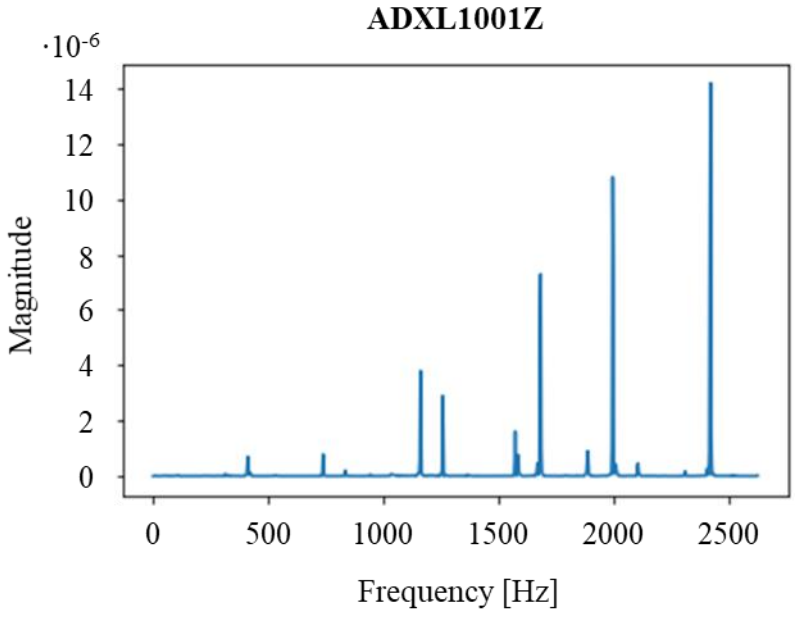

Figure 3.4: Measured Vibration Signal of the ADXL1001Z in the Frequency Domain

This experiment shows that a high sensitivity is necessary to distinguish the different frequency components of a vibration signal properly.

Moreover the comparison of both measured signals in the frequency domain illustrates the influence of the spectral noise level. The accelerometer ADXL203EB has a spectral noise of 110 μg/(Hz)$^{0.5}$ and the ADXL1001Z has a spectral noise of 30 μg/(Hz)$^{0.5}$ (compare Table 2). It can be seen that the vibration signal of the ADXL1001Z does not oscillate and that is has exact peaks. Instead of this, the spectrum of the ADXL203EB has an oscillation in the signal and the peaks are not as sharp as in Figure 3.4. In spite of the spectral noise level of the ADXL203EB, the peaks can be detected easily. Moreover, this accelerometer can measure the vibration in two different axis; this is important to demonstrate that both axes output the same vibration signal. Therefore, accelerometer ADXL203EB is used for measuring the vibration signal generated by rolling element bearings. Besides these physical properties, the price of $ 36.26 is also appropriate for a large amount of vibration measurement boxes.

As shown in Figure 3.3, the noise level increases above 2 kHz, and above 2.3 kHz it is significantly higher than at frequencies lower than 2 kHz. This higher noise level is caused by the bandwidth of the accelerometer ADXL203EB, which is limited up to 2.5 kHz (as shown in Table 2). Hence, the bandwidth has a limiting influence on the accelerometer measurements.

This drawback of the ADXL203EB has to be considered during the experiments and the vibration measurements, since the frequencies that can be measured are in the range between 0 Hz and approximately 2.5 kHz. At least three harmonics of the defect

frequencies have to be measured to be verify that there is a signal and not just noise. To distinguish a signal from noise is necessary to ensure the detection of rolling element bearing defects. Therefore, the frequency of the third defect harmonic has to be lower than 2.5 kHz. As explained in Chapter 2.2, the defect frequencies BPFI, BPFO, BSF and FTF are dependent on the rolling element bearing characteristics and the spindle turning frequency $\omega_{Spindle}$. The spindle turning frequency is the only parameter that is adjustable, so an appropriate spindle turning frequency has to be chosen. This appropriate spindle turning frequency and hence the rotations per minute (RPM) of the spindle can be calculated. According to Chapter 2.2, the highest defect frequency is the frequency caused by a defect on the inner race (BPFI). Equation (1) in Chapter 2.2 can be converted to equation (10):

$$RPM_{max} = \frac{2 \cdot BPFI}{3 \cdot n \cdot \left(1 + \frac{d_B}{d_P} \cdot \cos(\theta)\right)} \tag{10}$$

The factor 3 in the divisor is caused by the requirement that the third harmonic of the defect frequency must be measured. Since the bandwidth of the accelerometer is the limit of the vibration measurements, the frequency of the third harmonic of the defect frequency has to be lower than the bandwidth. As an increasing noise level can be seen at frequencies above 2 kHz in Figure 3.3, the limit of the accelerometer ADXL203 is set to 2 kHz. By converting this maximum frequency to RPM, equation (11) can be derived from equation (10):

$$RPM_{max} = \frac{2 \cdot 2000Hz \cdot 60s/min}{3 \cdot n \cdot \left(1 + \frac{d_B}{d_P} \cdot \cos(\theta)\right)} \tag{11}$$

If equation (11) is considered for the experiments and the vibration measurements, the vibration measurement box can detect all possible defects of the rolling element bearings.

The maximum measured frequency of the accelerometer ADXL203EB is not much higher than the limits of velocity transducers. However, the advantage is that the ADXL203EB emphasizes the higher frequencies and that the whole system will be based upon one accelerometer. If a customer requires an accelerometer with a higher bandwidth, it can be purchased and integrated in the vibration measurement box, since all the vibration analysis tools will be able to analyze vibration data based on acceleration. Therefore, a solution with an accelerometer is more flexible.

3.1.2 Microcontroller Teensy 3.2

The accelerometer measurements must be sampled and stored in order to analyze the data subsequently. Therefore, the two MCUs Teensy 3.2 and the Particle Photon are considered to sample the acceleration.

Teensy 3.2 Particle Photon

Figure 3.5: Illustration of the Teensy 3.2 and the Particle Photon [68, 69]

An illustration of these MCUs is shown in Figure 3.5. The relevant characteristics of these MCUs are shown in Table 3 and are based upon the datasheets of the MCUs [68, 70].

Table 3: Characteristics of the Teensy 3.2 and the Particle Photon

Characteristics	Teensy 3.2	Particle Photon
Max. Sampling Rate	1 MHz	30 MHz
Storage	256 kbytes	1 Mbyte
# ADC	2	8
Price	$ 19.80	$ 19.00

An MCU is chosen for sampling the accelerometer measurements, since these MCUs have a real time clock (RTC) and their system is deterministic. With a non-deterministic time system, the sampling rate of the accelerometer and therefore, the time interval between samples can vary. To get exact results and precise time intervals between samples, a deterministic system has to be utilized. By using the Particle Photon or the Teensy 3.2, this is already achieved. With a SBC such as the Raspberry Pi or the BeagleBone Black a deterministic time system cannot be achieved easily. A deterministic system on the Raspberry Pi or the BeagleBone Black requires specific programs and a real time Linux Kernel. Therefore, an MCU is used for sampling the accelerometer.

The chosen accelerometer outputs an analog signal, so at least one ADC is needed. This is ensured by both MCUs, since the Teensy 3.2 has two and the Particle Photon has eight ADCs (see Table 3).

The accelerometer´s bandwidth is limited at 2.5 kHz, and the Nyquist frequency explained in Chapter 2.3.1 requires a sampling rate that is at least two times higher than the highest frequency component that can be measured. Hence, the sampling frequency has

to be higher than 5 kHz. Both presented MCUs achieve this. The ADC of the Teensy 3.2 has a maximum sampling rate of 1 MHz, and the Particle Photon can even sample with 30 MHz (see Table 3).

To be able to sample with a frequency higher than 5 kHz, there has to be enough storage as well to store all these sampled data points on the MCUs. The Particle Photon has a flash memory of 1 MByte whereas the Teensy 3.2 has a flash memory of 256 kbytes. However, there is already used storage on the Particle Photon as mentioned in its datasheet [70]. Since the exact amount of flash memory that can be used is not explained, the free space was investigated by storing data points in hexadecimal format (three digits) on it until the firmware crashed. The Particle Photon is able to store 5250 samples in hexadecimal format without crashing. That means that the maximum sampling rate of the Particle Photon can be 5.25 kHz, if a vibration signal is sampled for one second. Moreover, each sampled data point has to be converted from decimal (six digits, e.g. -0.134) to hexadecimal to reach this sampling rate.

The same investigation was done with the Teensy 3.2. Although this MCU has less flash memory, it can store more than 6500 samples in decimal format (six digits). The exact storage limit was not investigated, but the Teensy 3.2 can store a larger amount of samples than the Particle Photon. Therefore, the Teensy 3.2 is chosen for sampling the accelerometer. The price of the Teensy 3.2 showed in Table 3 is almost the same as the price of the Particle Photon, so this was not considered for choosing the MCU.

3.1.3 SBC BeagleBone Black

Even though the MCU is used for sampling precisely the vibration signal, it cannot be used for giving feedback as to the rolling element bearing condition after the measured vibration signal is analyzed. This is because the Teensy 3.2 cannot be embedded into an IoT architecture, and it can just be controlled if it is connected to another gateway as for instance a computer or a SBC. Moreover, its computing power is not important enough for complex data analysis. Therefore, a SBC has to be chosen to control the MCU. The SBC can be embedded into an IoT architecture, so it can be controlled from anywhere. By choosing a SBC for controlling the MCU, it is also preferable to analyze the measured vibration signal with the SBC, since it is able to run different and higher programming languages (as for instance Python) compared to the Teensy 3.2 that can just run C.

The BealeBone Black wireless is chosen as the SBC with which the vibration signal is analyzed, since this SBC enables a WiFi connection and can be embedded easily into an IoT architecture. Compared to the Raspberry Pi, the BeagleBone Black is not that popular and has a lower processing power. It is still used for this vibration measurement box, since there is a developed board with which the Teensy 3.2 can easily be tied to the BeagleBone Black. This enables a compact construction and a better use of space (Figure 3.6). The lower processing power does not affect the analysis of the vibration signal, since this is an application that does not need a high processing power.

3.1.4 The assembled Vibration Measurement Box

The assembled vibration measurement box with the different hardware components is shown in Figure 3.6. Number 1 points to the accelerometer ADXL203EB, which is screwed tightly in a box that was manufactured for this purpose. A magnet is glued on the bottom of the box, so that the box can be easily mounted on a spindle. The accelerometer is connected with the MCU Teensy 3.2 by an M8 cable (number 2). The M8 cable has 4 different wires. These are used to connect the ground, the voltage and the x- and y-axis acceleration signal of the accelerometer to the MCU. The MCU is connected to the BeagleBoneBlack (number 5) by the green board (number 3). The BeagleBone Black is mounted on a DIN rail and gets powered by the power supply (number 4) with 5 V.

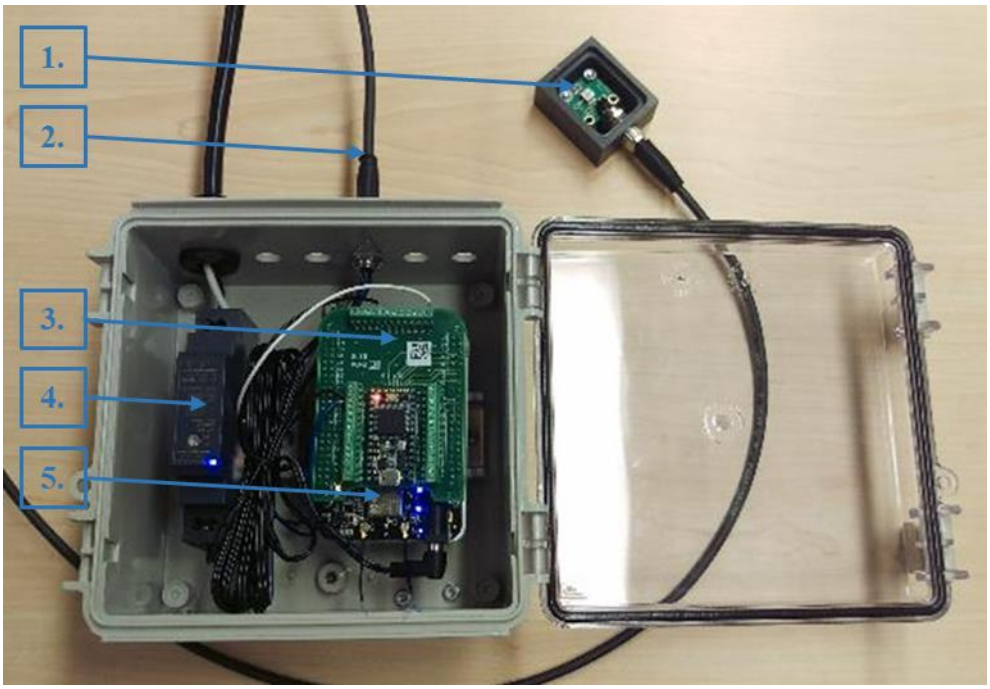

Figure 3.6: Assembled Vibration Measurement Box

The advantage of this vibration measurement box is its compact set-up and hence, the accelerometer can be easily mounted on a spindle with the magnet as can be seen in Figure 3.7.

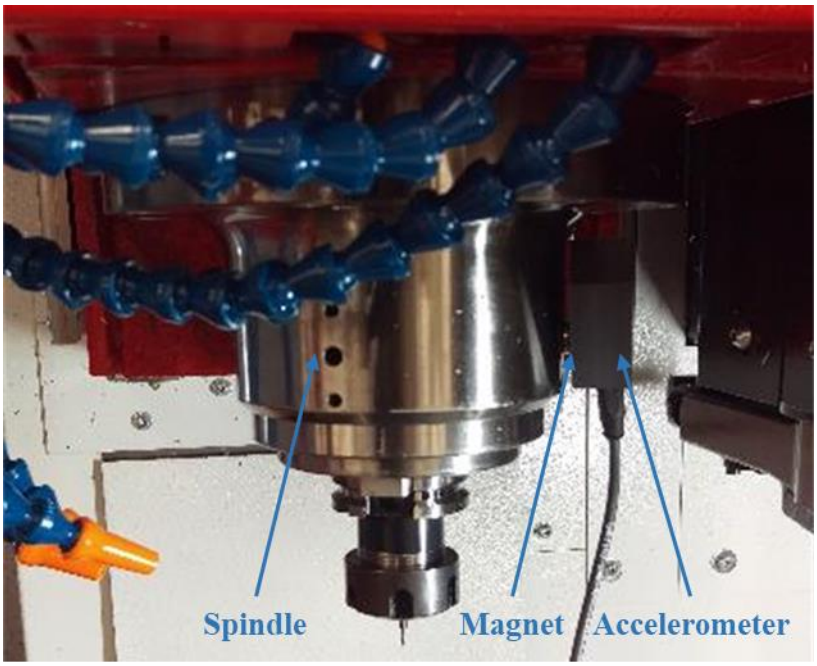

Figure 3.7: Attachment of the Accelerometer Box on the Spindle

Moreover, the box can be placed outside of the machine and hence, there cannot be any effects of the cooling or other process influences on the box. Only the accelerometer box has to be mounted in the machine. The drawback is the M8 cable that connects the accelerometer and the MCU, since it is in the working area (cf. Figure 3.8). If the machine has a rotating turret, a tool cannot be changed while the accelerometer is attached to the spindle. The accelerometer has to be put away from the spindle first. On the other hand, the M8 cable ensures that the accelerometer box is compact and does not need any WiFi module. This enables a stable connection between the MCU and the accelerometer, too.

Moreover, with the magnet it is possible to put the accelerometer quickly out of the working area of the machine.

Figure 3.8: Set-Up for Measuring the Acceleration

Overall, this vibration measurement box enables a less expensive acquisition and analysis of the vibration signal of the spindle. Moreover, it can be set up quickly and does not influence the working process, since the accelerometer box is compact and the M8 cable flexible. The risk of mounting the accelerometer box on the spindle by using a magnet is that the accelerometer box may itself vibrate and falsify the vibration signal. This is investigated in Chapter 4.

3.2 Proposed Software Components

Different software components are utilized to embed the vibration measurement box in the IoT architecture, to sample, to transfer and to analyze the measured vibration signal. An overview of this architecture is shown in Figure 3.9.

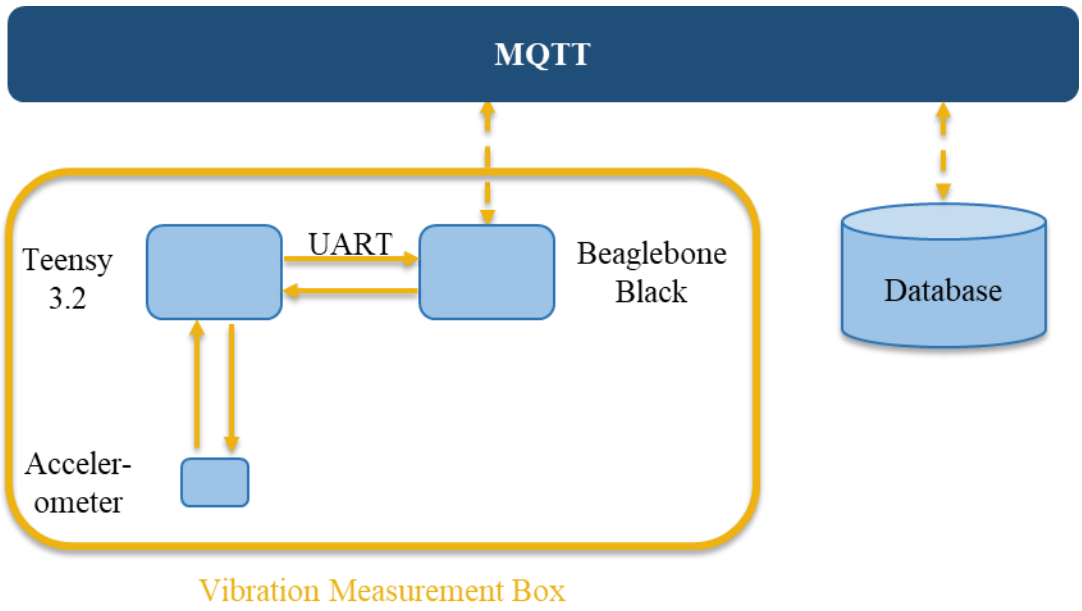

Figure 3.9: Used IoT Architecture and Communication Protocols

In the next sections, each step to sample, to transfer and to analyze the vibration signal is exemplified. Furthermore, the control of the vibration measurement box and the presentation of the results are illustrated.

3.2.1 Control of the Vibration Measurement Box

The most important part of the vibration measurement box is the BeagleBone Black and the software and programs it used. With the BeagleBone Black the command to sample a vibration signal, to transfer the sampled signal from the Teensy 3.2 to the

BeagleBone Black and to analyze the signal is placed. Moreover, the BeagleBone Black has a WiFi connection and publishes the result to the cloud by using the MQTT message broker.

Node-Red is installed on the BeagleBone Black and is used as a human machine interface (HMI). The reason is, that Node-Red is easy to use, is web-based and can tie the Teensy 3.2, the BeagleBone Black, analysis programs and the MQTT message broker together. All the following Node-Red flows can be executed and modified location-independent (cf. Chapter 2.4.2).

Before the sampling of the measured vibration signal can be initiated, the sampling parameters, sampling frequency and number of sampled data points have to be set. A schematic of setting these parameters is shown in Figure 3.10.

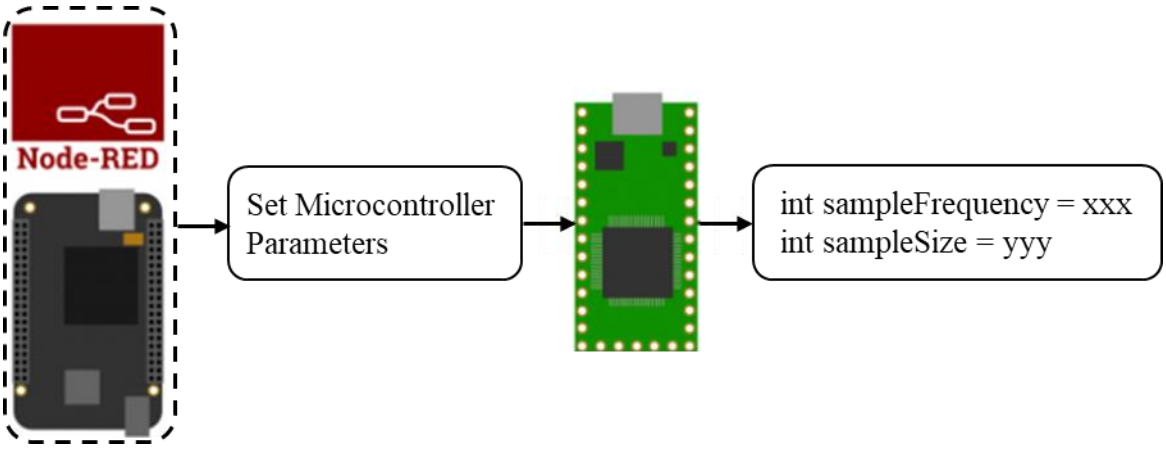

Figure 3.10: Schematic for Setting Sampling Parameters

The created Node-Red flow with which these parameters can be set is shown in Figure 3.11. This is the actual HMI, and the user can set the sampling parameters by clicking the blue button on the left side of the blue node. In the function node, the sampling

parameters can be defined by entering the sampling frequency in Hz and the sampling size in desired number of sampled data points.

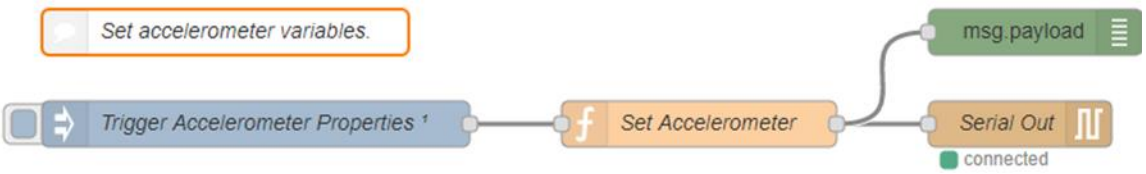

Figure 3.11: Node-Red flow for Setting the Sampling Parameters

The brown node on which is written *Serial Out* symbolizes the code for the UART connection between the BeagleBone Black and the Teensy 3.2. It transfers the sampling parameters to the Teensy 3.2. On the Teensy 3.2, a C-code runs and adjusts its sampling parameter by using the latest input. The green node *msg.payload* prints the sampling parameters on the debug window of Node-Red, so that the user can check if the right parameters were transmitted.

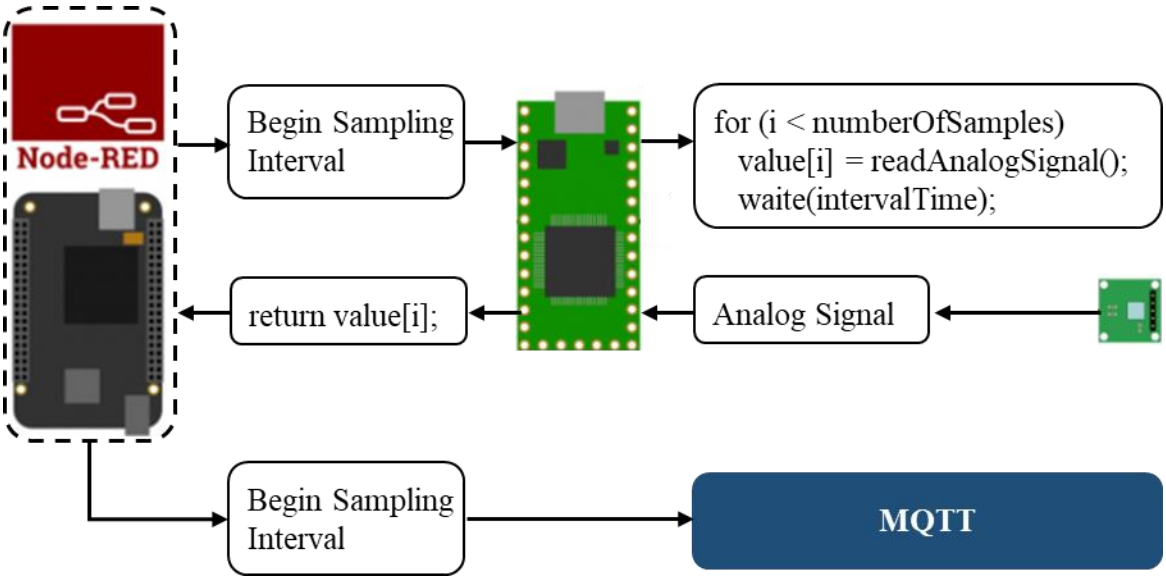

Figure 3.12: Schematic for Executing a Sampling Interval

Figure 3.12 represents the schematic for executing a sampling interval. At first, the sampling interval has to begin. This can be initiated by clicking on the blue button of the Node-Red flow that is shown in Figure 3.13. As presented in Chapter 2.2, the bearing characteristics as well as the spindle turning speed are necessary to calculate the defect frequencies. Therefore, by clicking the blue button, the condition data as the bearing characteristics, the spindle turning speed and the sampling rate are stored on the BeagleBone Black.

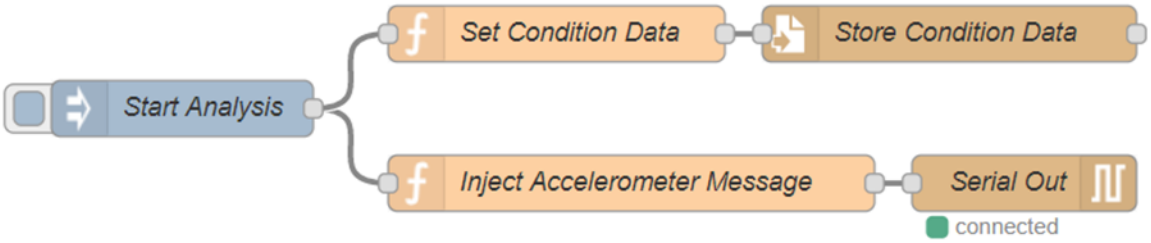

Figure 3.13: Node-Red Flow for Beginning the Sampling Interval

The Teensy 3.2 starts sampling the measured vibration signal and stores all the sampled data points until the sampling interval is finished. This code programmed in C is illustrated in the appendix A.1. Afterwards, the values are transmitted to the BeagleBone Black by using the same UART connection as for the beginning the sampling interval (cf. Figure 3.14).

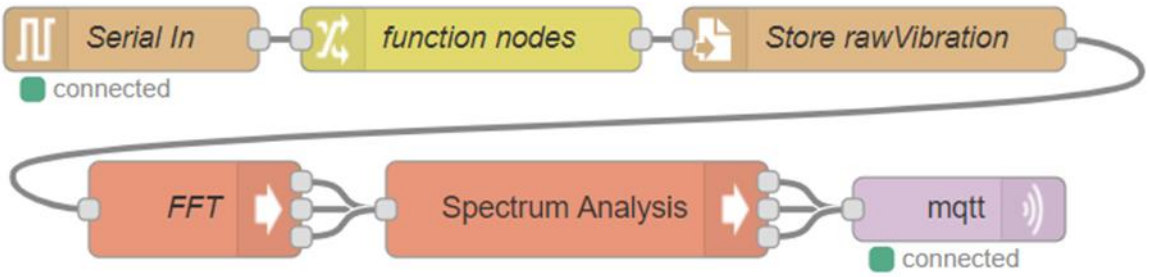

Figure 3.14: Node-Red Flow for Receiving and Processing the Sampled Data

There are several function nodes in this flow with which the received data get structured and cleaned. The cleaned data are subsequently stored as a text file on the BeagleBone Black. By storing the raw vibration signal, the user of this vibration measurement box can get the raw data, if a closer look is required. Moreover, it is an efficient way to transfer as many data points as required to the analysis tool, which is programmed with Python (see Chapter 3.2.2). The red nodes are execution nodes with which the Python codes get executed. The first node performs the FFT, the second one the analysis of the spectrum. It is also possible to pass the raw vibration data directly to these Python codes by using the execution nodes. However, only a limited number of data points can be transferred to the program on this way.

After the vibration signal is analyzed and the Python code gave feedback as to the bearing condition, this feedback is published to MQTT (purple node) with the Node-Red flow shown in Figure 3.14. As to Figure 3.9, this feedback is then stored in a database and can be looked up location-independent. Moreover, this feedback can be used for other web-based applications as for instance dashboards. This web-based application is not part of this Thesis, though.

3.2.2 Analysis of the Vibration Signal

Different analysis techniques as well as their advantages and drawbacks are explained in Chapter 2.3.2. Considering these advantages and drawbacks, the spectrum analysis is chosen for analyzing the vibration signal. As mentioned in Chapter 2.3.2, the analysis techniques of the time domain can be used for trending the rolling element bearing health in general, but these techniques have their limitations and therefore, it is not possible

to detect the particular defect and to distinguish this defect from other vibration signals. Moreover, the cepstrum analysis in the frequency domain is not used, since it is aimed at defects with sidebands. These are BPFI and BSF defects, but BPFO defects cannot be detected. The waterfall analysis is a powerful technique in the frequency domain, but it is not used for the purpose of this Thesis, since multiple measurements with different spindle turning frequencies are needed. In this Thesis, just one measured vibration signal shall be analyzed. However, the waterfall analysis is a great analysis technique if the machine has to be investigated more in detail.

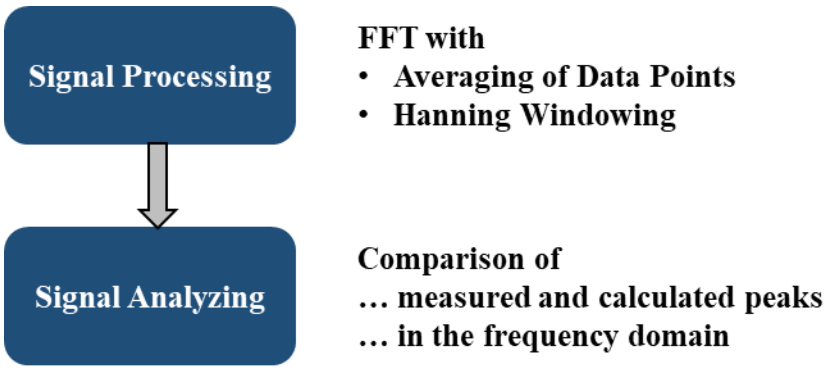

Figure 3.15: Procedure of Processing and Analyzing the Vibration Signal

Considering the explanations of Chapter 2.3.2, the procedure illustrated in Figure 3.15 is implemented to get a vibration signal with a high quality in the frequency domain and reliable results.

At first, the sampled vibration signal is converted in the frequency domain by using the FFT. This is performed by averaging data points together and using a Hanning window. Averaging data points together reduces the noise and generates a better signal representation in the frequency domain. This data averaging is undertaken by breaking the time-series data into segments that then get overlapped. By this overlapping, the data points

get averaged. After this the Hanning window is used for converting the signal from the time domain in the frequency domain. The Hanning window is used, since it suits best for vibration signals, reduces the effects of leakage and has a good frequency resolution (cf. Chapter 2.3.2).

```
    # Force the overlap to be one-half of the window length
    nOverlap = nPerSeg // 2

    # This code breaks the input data into a set of segments
    step = nPerSeg - nOverlap
    shape = inputData.shape[:-1]\
          + ((inputData.shape[-1]-nOverlap)//step, nPerSeg)
    strides = inputData.strides[:-1]\
            + (step*inputData.strides[-1], inputData.strides[-1])

    inputData = np.lib.stride_tricks.as_strided(inputData, shape=shape,
                                                strides=strides)

else:

    nPerSeg = inputData.shape[0]

# Create a hanning window
window = np.hanning(nPerSeg)

# Create the windowed FFT magnitudes
result = inputData - np.expand_dims(np.mean(inputData,axis=-1),axis=-1)
result = window * result
```

Figure 3.16: Segment of the Code to perform the FFT

Figure 3.16 represents the code with which the data in the time domain get overlapped and which applies the Hanning window to perform the FFT. The complete code is shown in the appendix A.2.

The data of the vibration signal in the frequency domain are stored as a text file on the BeagleBone Black after the FFT is performed. This enables the user to get direct access to these data if needed. A typical spectrum of measurements is shown in Figure 3.17.

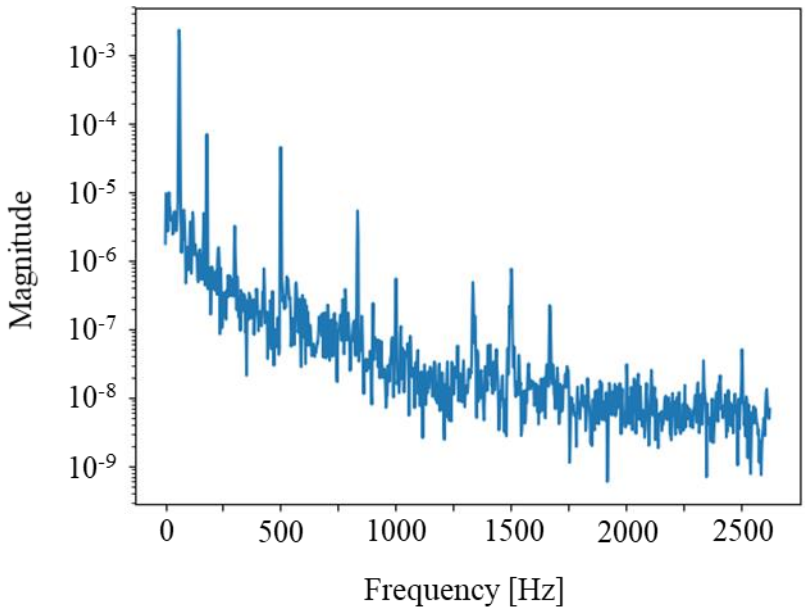

Figure 3.17: Spectrum of a measured Vibration Signal

The next step is to analyze the vibration signal in the frequency domain. The procedure of this analysis is illustrated in Figure 3.18. The code with which this analysis is realized is presented in the appendix A.3. The spectra generated by using the FFT have a slope (cf. Figure 3.17). Therefore, the spectrum is broken into frequency segments to be able to detect the highest points of each frequency segment. Otherwise, the highest points would all be found in the lower frequency range (f < 500 Hz), since the slope emphasizes the amplitude of the lower frequencies. In each segment the ten highest amplitudes are considered and it is checked, if these are close together (± 5 Hz). If this is the case, just the frequency with the highest amplitude is considered, and the remaining frequencies in this range are deleted.

In order that these detected frequencies can be compared with the calculated frequencies that are mentioned in Chapter 2.2, the bearing geometry and the used spindle turning speed (in RPM) are used to calculate the frequencies of the defect and the spindle.

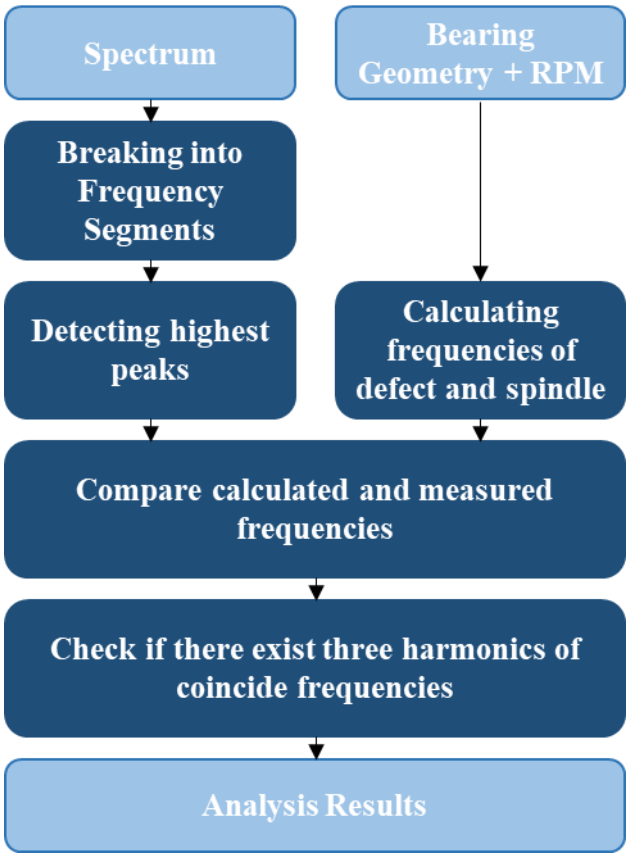

Figure 3.18: Procedure of Analyzing the generated Spectrum

The calculated frequencies and the detected frequencies with the highest peaks are subsequently compared, at which an allowable error of 3.5 % is considered. The reason is that in the equations to calculate the defect frequencies, no slip is included (cf. Chapter 2.2).

At this juncture, the number of harmonics of the coincide frequencies is checked. If there exist at least three harmonics of one type of frequency (e.g. spindle frequency), it is assumed that this characteristic frequency is detected. It can be seen in Chapter 4 that three harmonics are enough to detect a signal.

The analysis result is the output of the code and specifies the detected frequency. If the bearing is healthy, just the spindle frequency is detected and just the output *Detect Spindle Harmonics* appears. If an inner race defect is detected, *Detect BPFI Defect* is the output. Accordingly, the output is *Detect BPFO Defect* or *Detect BSF Defect* if an outer race defect or rolling element defect is detected. A cage defect cannot be detected with this code, since its characteristic FTF frequency is quite small and has many coincidences with the measured frequencies, even if there is no cage defect. Moreover, a cage defect is very rare and is usually a result of another defect that occurred earlier.

CHAPTER 4. EXPERIMENTS AND RESULTS

The vibration measurement box is tested on three different CNC milling machines with different characteristics, different bearings and different spindle turning speeds. The first machine is an EMCOMILL E350 manufactured by EMCO GmbH, the second machine is a Bridgeport V480 of Hardinge Inc. The last CNC milling machine is a GROB G515 that is produced by GROB-WERKE GmbH & Co. KG. Besides these milling operations, experiments were performed during a drilling operation on a machine of the EX-CELL-O Machine Tools Inc.

4.1 EMCOMILL E350

The EMCOMILL E350 is a three axis CNC milling machine that has a speed range from 50 to 10,000 RPM and is based upon a Siemens 828D controller. A representation is shown in Figure 4.1.

Figure 4.1: EMCOMILL E350 [71]

This machine is located in the Montgomery Machining Mall of the George W. Woodruff School of Mechanical Engineering and is chosen, because it is not frequently used and most likely this machine does not have any defects. The spindle utilizes a NSK rolling element bearing with the part number 45BER10STV1VSUELP3 R MTSX5. This bearing has a nominal bore diameter of 45 mm, an outer diameter of 75 mm, 22 balls each with a diameter of 6.35 mm. The contact angle is 25°.

On this machine three different experiments were performed. The first and the second one were performed without any cutting and with different spindle turning speeds. The third one was performed when a working piece was being cut. During all experiments, the sensor was placed directly on the spindle as can be seen in Figure 3.7 and Figure 3.8.

4.1.1 Experiment 1 and 2: No Cutting, 3,000 RPM and 10,000 RPM

This experiment was performed without any cutting and with a spindle turning speed of 3,000 RPM. The vibration was measured and converted into the frequency domain. This spectrum is shown in Figure 4.2.

The analysis tool of the vibration measurement box could not detect any characteristic frequency, neither of the spindle frequency nor of a defect frequency. There are no harmonics and only noise presented in the spectrum. The same result occurred if the spindle turning speed was increased to 10,000 RPM.

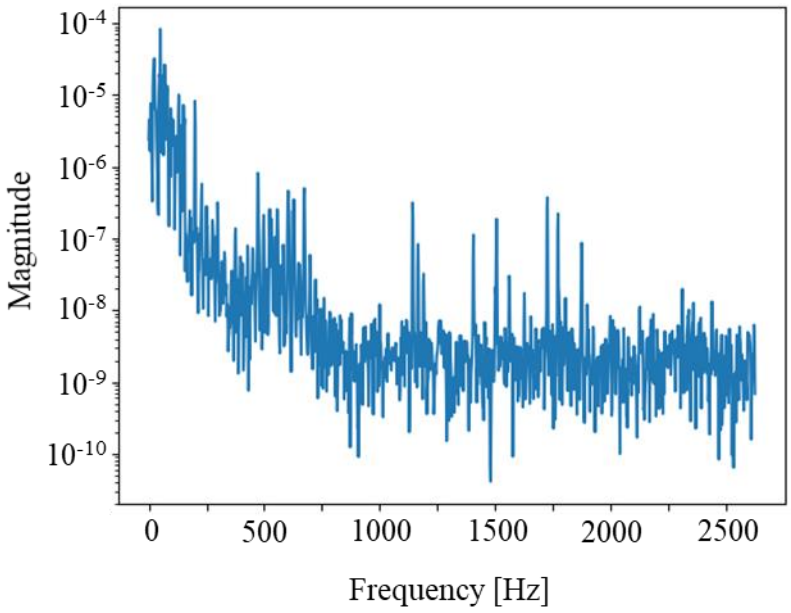

Figure 4.2: Spectrum of the Vibration Signal of Experiment 1 (EMCOMILL E350)

These experiments without cutting and different spindle turning speeds point out that the generated spectrum is too noisy and harmonics cannot be detected by the vibration measurement box.

4.1.2 Experiment 3: Cutting, 10,000 RPM

For the third experiment, the spindle turning speed was set to 10,000 RPM and a working piece was cut with a three flute, 0.5 inch diameter carbide end mill. The spectrum is presented in Figure 4.3.

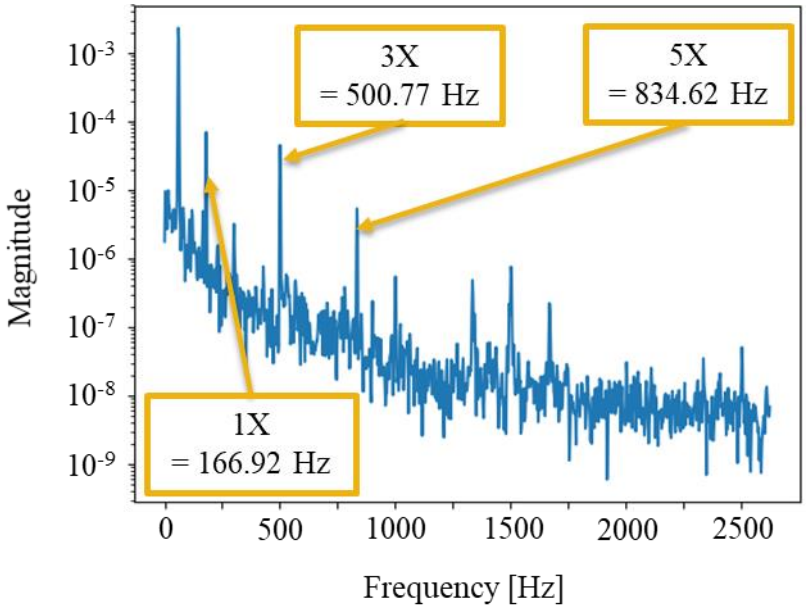

Figure 4.3: Spectrum of the Vibration Signal of Experiment 3 (EMCOMILL E350)

This spectrum is significant different from the spectrum shown in Figure 4.2. The magnitude is higher and a clear signal is presented. The output of the vibration measurement box was *Detect Spindle Harmonics.* Hence, the vibration measurement box detected spindle harmonics and no defect frequency. This means, the bearing of the EMCOMILL E350 is healthy.

If the spectrum is investigated in detail, this result is confirmed. Three harmonics (1X, 3X, 5X) of the spindle frequency are pointed out in Figure 4.3. Since the spindle turning speed was 10,000 RPM, its frequency $\omega_{Spindle}$ was 166.67 Hz. This coincides with the first detected peak that has a frequency of 166.92 Hz. The error between these two frequencies is 0.15 % and hence smaller than the accepted error of 3.5 %. Also the next two peaks in the spectrum coincide with the calculated frequencies of the third (3X = 500.77 Hz) and the fifth harmonic (5X = 833.35 Hz).

The question why only the first, third and fifth harmonics can be seen in the spectrum, whereas no second or fourth harmonics are presented, could not be answered completely. Probably, the tool and its number of flutes influence the vibration signal, since the number of vibration shocks that are indicated per rotation is dependent on the number of flutes.

The highest peak of the spectrum that has a magnitude higher than 10^{-3} is exactly at 60 Hz. There was already a peak at this frequency in Figure 3.3. This frequency is caused by electrical noise and indicates, that the vibration measurement box has to be grounded to avoid this peak. However, it does not falsify the results if the spindle is not turned with 3,600 RPM corresponding to 60 Hz.

The influence of the acceleration, velocity or displacement level on the representation of the vibration signal in the spectrum was investigated. The spectrum was first filtered by using a bandpass filter; subsequently the zero mean of the vibration signal was generated. In the end, Simpson´s rule was applied to integrate the signal. This procedure is shown in Figure 4.4.

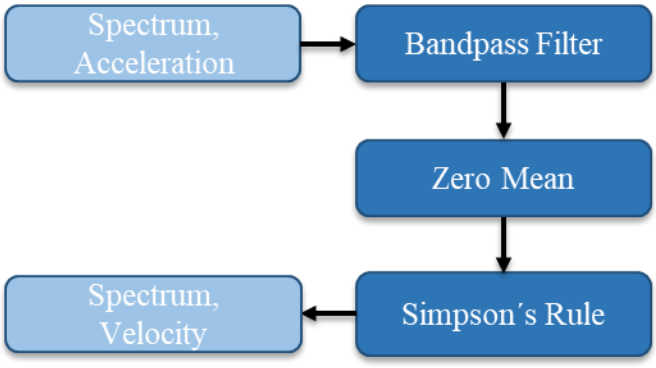

Figure 4.4: Procedure to integrate the Vibration Signal in the Frequency Domain

The bandpass filter is used to separate the frequency range that actually shall be integrated from frequencies that are lower or higher. Lower frequencies cause a drift of data and higher frequencies generate noise. The mean has to be set to zero so that the integration works properly. The Simpson´s rule is applied for integrating the vibration signal, since it uses parabolas and hence, it gives the exact area under discrete data points for constant, linear, quadratic and cubic functions. So the Simpson´s rule can be applied on most of the functions.

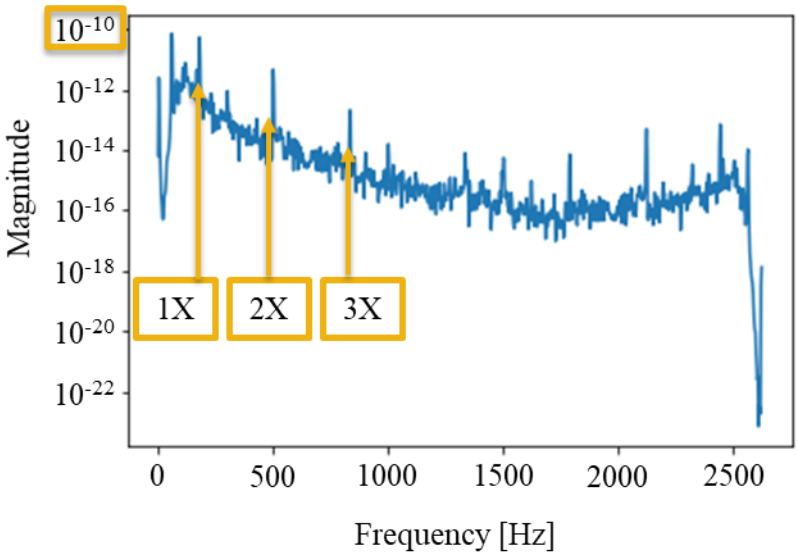

Figure 4.5: Spectrum of the Vibration Signal on the Velocity Level

Figure 4.5 represents the spectrum of the measured vibration signal after it was integrated to velocity. The magnitudes are a factor of 10^{-7} smaller than on the acceleration level. The overall noise level is smaller, too, but the magnitude difference between the peaks and the noise gets also smaller. Besides this, the amplitudes in the higher frequency range (> 1500 Hz) get emphasized and in the frequency range between 1000 and 1500 Hz the peaks are more difficult to distinguish from the noise. This effect increases if the signal

is further integrated on the displacement level. Since the original signal is changed and differently emphasized, the spectrum on the acceleration level is analyzed for further experiments.

4.2 Bridgeport V480

The Bridgeport V480 is a three axis CNC milling machine and has a speed range up to 10,000 RPM. It is based upon a Mitsubishi M70 Controller and is used for the next experiments due to a crash that happened. This machine is located at the Montgomery Machining Mall and one of the most used machines in this mall. The technician of this machine hears different sounds since the machine crashed. This sound could indicate a bearing defect.

Figure 4.6: Bridgeport V480 [72]

The different experiments were performed during cutting operations and with spindle turning speeds of 10,000 RPM and 7,640 RPM. The built in rolling element bearing is from NSK and has the part number Z0193 6208PSN24T1XVVC3. Its nominal bore diameter is

40 mm, the outer diameter 80 mm, it has 9 balls with a ball diameter of 11.906 mm and a contact angle of 0°.

4.2.1 Experiment 1: 10,000 RPM

This experiment was performed during a cutting operation and a spindle turning speed of 10,000 RPM. The vibration measurement box indicates a rolling element defect (BSF) and spindle harmonics in the vibration signal of this machine.

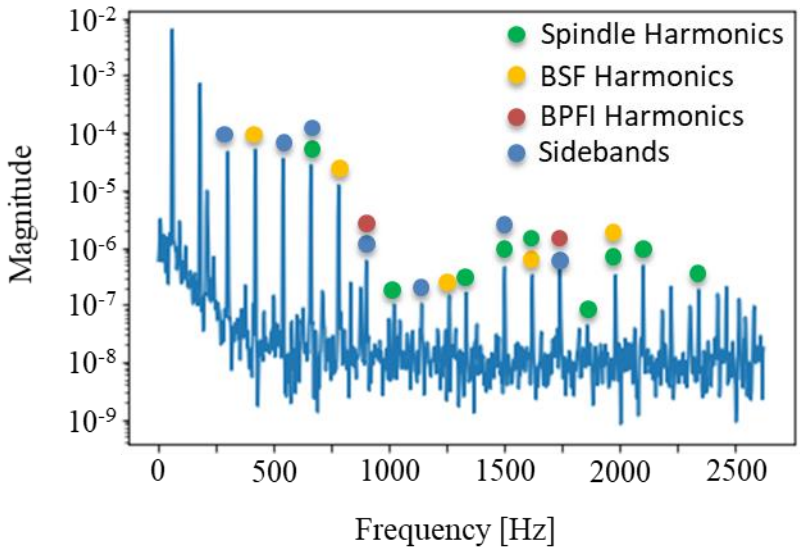

Figure 4.7: Spectrum of the Vibration Signal of Experiment 1 (Bridgeport V480)

A closer look at the spectrum of the vibration signal illustrates that the analysis tool of the vibration measurement box works properly. There are three peaks that have the same frequency as the BSF harmonics. Moreover, the typical sidebands of rolling element defects can be seen in the spectrum. There exist also more than three harmonics of the spindle frequency, so this is indicated. The red dots in the spectrum show that two peaks have the same frequency as the BPFI harmonics. Since there are just two harmonics, a

defect is not indicated by the vibration measurement box. However, these BPFI harmonics have the same frequency as two sidebands of the BSF defect. Therefore, a second experiment with a different spindle turning speed was performed to check if there actually could exist a BPFI defect. Besides this, several measurements were executed to check if the measured vibration signal changes. Every measured signal led to almost the same spectrum as illustrated in Figure 4.7. This indicates that the measurement works properly and the actual vibration signal of the spindle was measured.

4.2.2 Experiment 2: 7,640 RPM

This experiment was performed during a cutting operation and a spindle turning speed of 7,640 RPM. Again, the vibration measurement box indicates a rolling element defect (BSF) and spindle harmonics in the vibration signal of this machine.

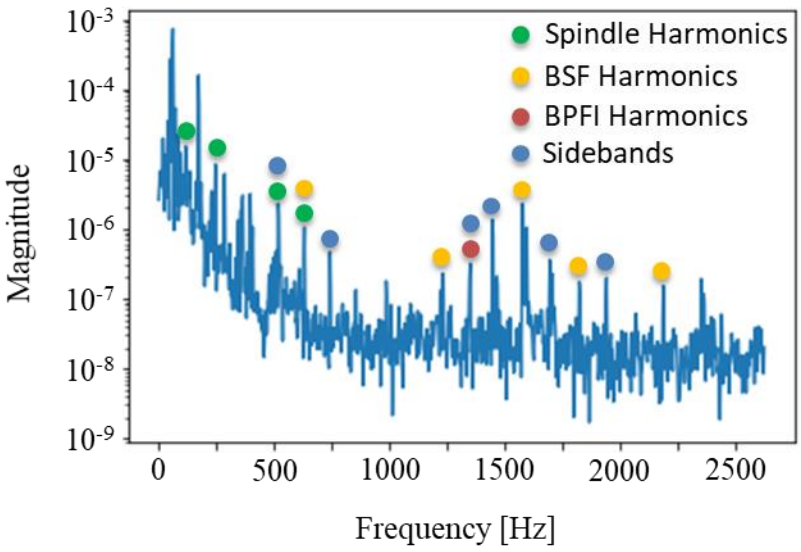

Figure 4.8: Spectrum of the Vibration Signal of Experiment 2 (Bridgeport V480)

Figure 4.8 confirms the detection of a rolling element defect and spindle harmonics. This experiment also indicates that there is no defect on the inner race of the rolling element bearing. There exists again one coincidence between the calculated BPFI and one detected peak. Since there exist no other accords and this one peak is also indicated to be a sideband of the BSF harmonic, this bearing has most likely no defect on the inner race.

Unfortunately, the spindle of this machine cannot be disassembled to check if the bearing actually has a rolling element defect. These experiments show that the vibration measurement box works properly, though. Each measured signal changes the frequencies of the measured peaks according to the spindle turning speed. Moreover, the results of the analysis of these measured signals are correct, as a manual check with the spectrums shows.

4.3 GROB G515

The GROB G515 is a modular machining center that is located at a plant of the Ford Motor Company. This machine is frequently used and machining operations that shall be transferred in the production are tested on it. In the actual production line, the Ford Motor Company has also these GROB G515 machining centers. This machine is very new, so the vibration measurement box shall indicate spindle harmonics and shall not indicate any defect harmonics.

4.3.1 Experiment 1: 5,400 RPM

One experiment was performed on the GROB G515. A work piece that is also machined in the actual production was face milled with a spindle turning speed of 5,400 RPM.

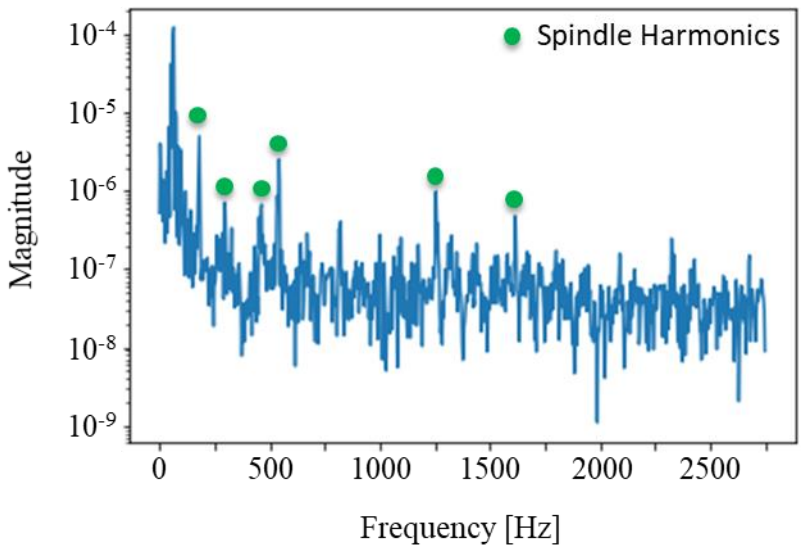

Figure 4.9: Spectrum of the Vibration Signal of Experiment 1 (GROB G515)

The vibration measurement box measured the vibration signal of the spindle and could only detect spindle harmonics. This is the result we expected. The analysis tool works properly, as can be seen in Figure 4.9. There are no other peaks beside the spindle harmonics.

4.4 EX-CELL-O

This machine is located at a plant of the Ford Motor Company and is used to check if the vibration measurement box can analyze vibration signals generated by drilling operations, too. This machine is old and the bearing configurations are not known. Therefore, the bearing health cannot be investigated. However, it can be tested if the spindle frequencies can be detected in the spectrum. This already indicates if the vibration measurement box works properly during drilling operations.

4.4.1 Experiment 1: 8,000 RPM

The vibration measurement box detects the spindle harmonics and a check of the spectrum confirms this result. The frequencies of the peaks coincide with the calculated spindle frequencies as it can be seen in the spectrum (Figure 4.10).

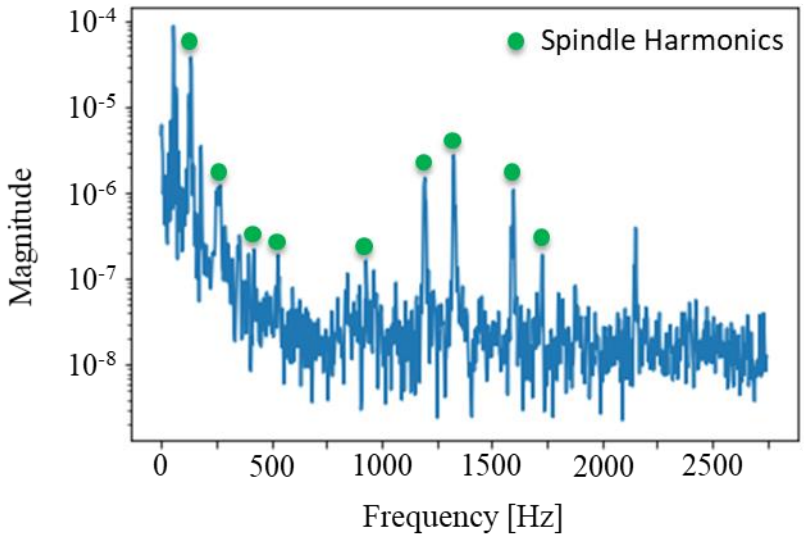

Figure 4.10: Spectrum of the Vibration Signal of Experiment 1 (EX-CELL-O)

Although the bearing characteristics are unknown, this experiment indicates that the vibration measurement box can be used for monitoring the bearing conditions during drilling operations, too.

4.5 Summary of the Results

The different experiments on the different machines and a comparison between the result of the vibration measurement box and a manual analysis indicate the following aspects:

- The vibration measurement box has to be used during cutting operations. This is confirmed during milling and drilling operations.
- The accelerometer measures the vibration signal properly.
- The FFT generates a spectrum with which the signal can be analyzed properly.
- The analysis of the spectrum detects defects and spindle harmonics automatically.
- The concern that the magnetic attachment of the accelerometer on the spindle could falsify the measured vibration signal (see Chapter 3.1.4) could be eliminated.
- There is always a peak at 60 Hz due to electrical noise.
- There can be random matches between different characteristic frequencies. Exemplary, two BPFI harmonics match with sidebands of the BSF harmonics in Figure 4.7.

To avoid having random matches influence the result of the vibration measurement box, this box has to find three harmonics to indicate a defect. Moreover, the spectrum and the raw vibration data are stored on the BeagleBone Black. With these data, a defect can be checked a second time, if it is necessary.

CHAPTER 5. CONCLUSION AND OUTLOOK

Concluding, the developed vibration measurement box and the acquired results of the experiments are examined critically. Based on this, an outlook which exemplifies possible or necessary further research is given.

5.1 Conclusion

This thesis presents the development of a device for automatic monitoring of rolling element bearing conditions. Based on the expressed purpose, the device is portable and enables analysis of rolling element bearings of different machines. The device is internet enabled and is embedded into an IoT architecture, whereby the analysis results are stored in a database. Moreover, the vibration measurement box can be controlled location-independent.

In the beginning, the characteristics of rolling element bearing defects as well as the advantages and drawbacks of different hardware and software components are explained in the state-of-the-art. These were considered to develop the vibration measurement box and to code the different algorithms to sample, to process and to analyze the measured vibration signal.

Subsequently, experiments on different machines with different rolling element bearings, different running conditions and different machining operations were performed. These experiments show that both the developed device as well as the coded algorithms work properly. Harmonics of spindle frequencies and bearing defects could be detected and the feedback as to the bearing conditions was given automatically. Thereby, time and

cost can be saved. This became especially apparent while proofing the accuracy of the feedback of the vibration measurement box. The manual analysis of the generated spectrums took a lot of more time and careful work was necessary to categorize each peak. The vibration measurement box simplifies this process significantly.

Additionally, the experiments point out that the vibration measurement box has to monitor the rolling element bearing condition during cutting operations. These cutting operations cause a clear vibration signal that can be analyzed properly. If the spindle turns without cutting a work piece, no signal could be distinguished from noise.

A limitation of this vibration measurement box is the accelerometer ADXL203EB that has a bandwidth up to 2.5 kHz. Vibration signals with a frequency higher than 2.5 kHz cannot be measured. Therefore, the spindle turning speed has to be adjusted to have at least three harmonics of the highest defect frequency (BPFI) within the bandwidth of the accelerometer.

The M8 cable that connects the accelerometer and the MCU is in the working area of the machine, so tool changes cannot be performed if the machine has a rotating turret. To change the tool, the accelerometer has to be removed from the spindle. Therefore, a magnet is used to attach the accelerometer to the spindle. This enables fast and simple attaching and removing. Moreover, the wired connection between the accelerometer and the MCU ensures the compact set-up of the accelerometer box. A wireless connection requires more components in the accelerometer box as for instance an additional MCU that provides a WiFi connection. The stability of this connection is dependent on the WiFi

quality, whereas the M8 cable ensures a deterministic and stable sampling of the accelerometer.

5.2 Outlook

Since the main limitation of the vibration measurement box is the bandwidth of the accelerometer, a high quality accelerometer as for instance the PCB Model 626B03 can be integrated to increase the bandwidth. The PCB Model 626B03 has a bandwidth up to 6 kHz. With such a high bandwidth, more harmonics of bearings can be detected without any constraints regarding the spindle turning speed.

Moreover, the spectral noise level (0.2 – 1.5 $\mu g/(Hz)^{0.5}$) is significantly lower than the spectral noise level of the used accelerometer ADXL203EB (110 $\mu g/(Hz)^{0.5}$. With this low spectral noise level, it could be possible to detect rolling element bearing defects or frequency harmonics without a cutting operation. It also simplifies the distinction between the contained signal and noise. Compared to the ADXL203EB, the PCB Model 626B03 costs $ 455.00 and needs a specific signal conditioner.

In all spectrums a peak at a frequency of 60 Hz is shown. This peak is generated by electric noise and can probably be eliminated by grounding the voltage supply of the SBC.

Unfortunately, it was not possible to disassemble a rolling element bearing whose condition was monitored. This could be a next step to continue this research.

Related to the programmed algorithms, the amplitudes of the measured peaks of defect frequencies can be tracked and can be compared with the amplitudes of the spindle harmonics. This indicates the severity of the bearing defects and enables the opportunity

to perform historical analysis and to create web-based dashboards that illustrate the severity of the bearing defects.

APPENDIX A. DESCRIPTION OF THE ALGORITHMS

This appendix illustrates the different algorithms that run on the Teensy 3.2 and the BeagleBone Black. Two algorithms are implemented on the BeagleBone Black. One algorithm describes the FFT, one analyzes the spectrum.

A.1 Algorithm on the Teensy 3.2

With this algorithm the accelerometer is sampled. The sampling rate and the number of collected data points can be adjusted by using an UART connection.

```
/*
 * Program for sampling acceleration
 * April 9, 2019
 * © Daniel Newman, Niklas Tritschler
 */
#define HWSERIAL Serial1
#define INPUT_SIZE 30

const String ACCELEROMETER = "Accelerometer";
const String CURRENT = "Current";
const String INTERVAL = "Interval";
const String ACCELCURR = "AccelCurr";
const String CURRENT1 = "Current-1";
const String CURRENT2 = "Current-2";
const String CURRENT3 = "Current-3";

const String START = "Start";
const String STOP = "Stop";
const String SET = "Set";

// Declare constants
uint16_t accelerometerSamplingRate = 2500;
uint16_t accelerometerSize = 2500;
uint16_t accelerometerLineSize = 500;

// Time between accelerometer samples in Hz
float sensorSamplingPeriod = 0.1;

uint16_t currentSamplingRate = 100;
uint16_t currentSize = 1000;
uint16_t currentLineSize = 500;

const int accelerometerX = 1;
const int accelerometerY = 0;
```

```
const int current1 = 8;
const int current2 = 7;
const int current3 = 9;

int intervalCount = 0;

const int ANALOG_READ_RESOLUTION = 8;
uint8_t accelerometerSample;
uint8_t currentSample;
uint8_t currentSample1;
uint8_t currentSample2;
uint8_t currentSample3;

bool sumCurrent = true;

const String SERIAL_DELIMITER = ",";
const String SERIAL_START = "";

// The Teensy supports up to 4 concurrent instances of IntervalTimer
IntervalTimer accelerometerSamplingTimer;
IntervalTimer accelerometerTimer;
IntervalTimer currentSamplingTimer;
IntervalTimer currentTimer;
IntervalTimer sensorIntervalTimer;

String currentOut = CURRENT;
String currentOut1 = CURRENT1;
String currentOut2 = CURRENT2;
String currentOut3 = CURRENT3;
String accelerometerOut = ACCELEROMETER;

String serialInput;

// Keep track of the current position for the accelerometer
uint16_t accelerometerCount = 0;
uint16_t accelerometerLineNumber = 0;

uint16_t currentCount = 0;
uint16_t currentLineNumber = 0;

uint8_t sensorSampleNumber = 10;

char DELIMITER = '|';

// Used to parse serial input commands
String dataId;
int samplingFrequency;
int numPoints;

// Setup function.
void setup() {
  HWSERIAL.begin(115200);
  analogReadResolution(ANALOG_READ_RESOLUTION);
}

// Main loop.
```

```
void loop() {

  while(HWSERIAL.available()) {

    // Get next command from Serial (add 1 for final 0)
    char input[INPUT_SIZE + 1];

    byte size = HWSERIAL.readBytes(input, INPUT_SIZE);

    // Add the final 0 to end the C string
    input[size] = 0;

    // Read each command pair
    char* command = strtok(input, "|");

    int position = 0;

    dataId = command;

    if(dataId == "StartAccelerometer"){
      accelerometerTimer.begin(sampleAccelerometer,1e6/accelerometerSamplingRate);
      break;
    }
    else if(dataId == START + CURRENT){
      currentTimer.begin(sampleCurrent,1e6/currentSamplingRate);
      break;
    }
    else if(dataId == START + "All"){
      accelerometerTimer.begin(sampleAccelerometer, 1e6/accelerometerSamplingRate);
      delay(100);
      currentTimer.begin(sampleCurrent,1e6/currentSamplingRate);
      break;
    }
    else if(dataId == START + INTERVAL){
      sensorIntervalTimer.begin(intervalCallback,1e6/sensorSamplingPeriod);
    }
    else if(dataId == STOP){
      sensorIntervalTimer.end();
      intervalCount = 0;
      break;
    }

    // Parse the serial input
    while (command != 0)
    {
      if(dataId == SET + ACCELEROMETER){
        switch(position){
          case 1:
            accelerometerSamplingRate = atoi(command);
            break;
          case 2:
            accelerometerSize = atoi(command);
            break;
        }
      }
      else if(dataId == SET + CURRENT){
```

```
        switch(position){
          case 1:
            currentSamplingRate = atoi(command);
            break;
          case 2:
            currentSize = atoi(command);
            break;
        }
      }
      else if(dataId == SET + INTERVAL){
        switch(position){
          case 1:
            sensorSamplingPeriod = atof(command);
            break;
          case 2:
            sensorSampleNumber = atoi(command);
            break;
        }
      }
      position++;
      command = strtok(0, "|");
    }
  }
}

void intervalCallback(){
  accelerometerTimer.begin(sampleAccelerometer, 1e6/accelerometerSamplingRate);
  delay(10);
  currentTimer.begin(sampleCurrent,1e6/currentSamplingRate);

  intervalCount++;

  if(intervalCount > sensorSampleNumber){
    sensorIntervalTimer.end();
    intervalCount = 0;
  }
}

char* mapToFloat(uint8_t value){
  // Using 5 Volts as the maximum ADC Value and 8 Bit ADC Resolution,
  // The normalized value is multiplied by 5/255. This simplifies to 1/51
  float returnVal = (value / 51.0)-2.5;

  char returnString[6];
//  returnString[5] = '\0';

  return dtostrf(returnVal,4,3,returnString);

}

// This callback is used to periodically return an accelerometer sample
void periodicSample(){
  accelerometerTimer.begin(sampleAccelerometer,1e6/accelerometerSamplingRate);
}

// This callback is used to quickly grab a series of accelerometer data points
```

```
// After this function completes one cycle, the timer is ended.
void sampleAccelerometer() {

  if (accelerometerCount + accelerometerLineNumber * accelerometerLineSize <
  accelerometerSize){
    accelerometerSample = analogRead(accelerometerX);

    accelerometerOut += mapToFloat(accelerometerSample);

    if(accelerometerCount >= accelerometerLineSize){
      HWSERIAL.println(accelerometerOut + SERIAL_DELIMITER);
//      accelerometerOut += "\n" + ACCELEROMETER + SERIAL_DELIMITER;
      accelerometerOut = ACCELEROMETER + ":";
      accelerometerLineNumber++;
      accelerometerCount = 0;
    }
    else{
      accelerometerOut += SERIAL_DELIMITER;
      accelerometerCount++;
    }
  }
  else{
    HWSERIAL.println(accelerometerOut + "END");
    accelerometerCount = 0;
    accelerometerLineNumber = 0;
    accelerometerOut = ACCELEROMETER + ":";
    accelerometerTimer.end();
  }
}

// samplingTimer callback function.
void sampleCurrent() {

  if (currentCount + currentLineNumber * currentLineSize < currentSize){
    currentSample1 = analogRead(current1);
    currentSample2 = analogRead(current2);
    currentSample3 = analogRead(current3);

    currentSample = currentSample1 + currentSample2 + currentSample3;
    currentOut += mapToFloat(currentSample);
    currentCount++;

    if(currentCount > currentLineSize){
      HWSERIAL.println(currentOut + SERIAL_DELIMITER);
      currentOut = CURRENT;
//      currentOut += SERIAL_DELIMITER + '\n' + CURRENT;
      currentLineNumber++;
      currentCount = 0;
    }
    else{
      currentOut += SERIAL_DELIMITER;
    }
  }
  else{
    HWSERIAL.println(currentOut + "END");
    currentCount = 0;
    currentLineNumber = 0;
    currentOut = CURRENT;
    currentTimer.end();
  }
}
```

A.2 Algorithm to Execute the FFT

This algorithm transforms the measured vibration signal from the time domain in the frequency domain. Data points of the time domain are averaged together and the Hanning window is applied.

```
'''
# Algorithm for FFT with Hanning Window and data point
# averaging
# Created by Niklas Tritschler
# Copyright © 2019. All rights reserved.
'''

import numpy as np
import warnings

# Getting vibrationData from .txt file
filepath_1 = '/home/debian/my_data/rawVibrationData.txt'

with open(filepath_1) as f:
    content = f.readlines()

# Removing the whitespace characters at the end of each line
content = [x.strip() for x in content]

#vibData = np.zeros(len(content))
vibData = np.zeros(1)

# Putting all the lines together in one array
for i in range(0, len(content)):
    rawData = content[i].split(":")
    vib = rawData
    vibData = np.append(vibData, vib)

# Cleaning arrray
index1 = np.argwhere(vibData == "Accelerometer")
vibData = np.delete(vibData, index1)
index2 = np.argwhere(vibData == "")
vibData = np.delete(vibData, index2)
vibData = np.delete(vibData, 0)

# Converting 2D array in 1D array
vibrationData = np.zeros(1)

for i in range(0, len(vibData)):
    for j in range(0, len(vibData[i])-6,6):
```

```
        vib = vibData[i][j] + vibData[i][j+1] + vibData[i][j+2]\
        + vibData[i][j+3] + vibData[i][j+4]
        vibrationData = np.append(vibrationData, vib)

vibrationData = np.delete(vibrationData, 0)

filepath_2 = '/home/debian/my_data/samplingRate.txt'

f2 = open(filepath_2, 'r')
samplingRate = f2.readline()
f2 = f2.close()

samplingRate = int(samplingRate)

# **************************Fill in storing vibrationData


# Creating FFT
# Created by Daniel Newman

# Setting parameter
minResolution = 5
pointsBetween = 5
maxSegments = 30
averaged = True

NyquistFreq = 0.5 * samplingRate

# Converting data from string to float
inputData = np.array(vibrationData).astype(np.float)

''' Function to get the FFT for a response
#
# Adapted from Scipy's Welch's Method function:
# https://github.com/scipy/scipy/blob/v1.1.0/scipy/signal/
# spectral.py#L286-L427
#
# Input:
#   inputData - reference signal we wish to analyze
#   minResolution - The minimum distance between two spectral
#                   peaks
#   pointsBetween - The number of bins between the minimal
#                   spectral peaks
#   maxSegments - The maximum number of segments to divide
#                 the data into
#   averaged - Dictates whether the various segments should
#              be averaged together
#
# Output:
#   fftFreq = an array of the freqs used in the FFT
#   fftMag = an array of the amplitude of the FFT at each freq
#            in fftFreq
#
#####################################################################
'''

# Force the inputData to be two-dimensional
inputData = np.atleast_2d(np.asarray(inputData))
if inputData.shape[0] > 1 and inputData.shape[1] > 1 :
```

```
# Force the inputData to be two-dimensional
inputData = np.atleast_2d(np.asarray(inputData))
if inputData.shape[0] > 1 and inputData.shape[1] > 1 :
    raise ValueError("inputData must be a column or a line vector")
    # Looking if the array is only along one axis

    # If we're looking at multiple samples, we should ensure that the
    # shape of the array is correct.
if inputData.shape[0] > inputData.shape[1]:
    inputData = inputData.T

# Get the length of the dataset
numPoints = inputData.shape[1]

# If we only have one row of data
if inputData.shape[0] == 1:
    # Make sure the number of points per segment is sufficient to
    # get a desired resolution. If we have a lot of data, limit
    # the number of segments because >30 has no meaningful effect
    # on variance reduction.
    if numPoints < maxSegments * NyquistFreq:
        nPerSeg = int(np.ceil(samplingRate \
                * pointsBetween / minResolution))
    else:
        nPerSeg = int(numPoints // (maxSegments/2))

    if nPerSeg > numPoints//2:

        nPerSeg = numPoints//2

        warnings.warn('The desired resolution cannot be achieved\n\
            due to limited available data. Consider reducing the\n\
            desired resolution or increasing the sampling time.',\
            UserWarning)

    # Force the overlap to be one-half of the window length
    nOverlap = nPerSeg // 2

    # This code breaks the input data into a set of segments
    step = nPerSeg - nOverlap
    shape = inputData.shape[:-1]\
          + ((inputData.shape[-1]-nOverlap)//step, nPerSeg)
    strides = inputData.strides[:-1]\
            + (step*inputData.strides[-1], inputData.strides[-1])

    inputData = np.lib.stride_tricks.as_strided(inputData, shape=shape,
                                        strides=strides)

else:

    nPerSeg = inputData.shape[0]

# Create a hanning window
window = np.hanning(nPerSeg)

# Create the windowed FFT magnitudes
result = inputData - np.expand_dims(np.mean(inputData,axis=-1),axis=-1)
result = window * result
```

```
FFTMag = np.fft.rfft(result, n=nPerSeg)
FFTMag = np.conjugate(FFTMag) * FFTMag

# Do stuff to scale the magnitudes.
# Review the original source for insight.
if nPerSeg % 2:
    result[..., 1:] *= 2
else:
    # Last point is unpaired Nyquist freq point, don't double
    result[..., 1:-1] *= 2

scale = 1.0 / (samplingRate * (window*window).sum())
FFTMag *= scale

FFTMag = np.rollaxis(FFTMag, -1, -2)

# Get the frequencies for the FFT
#print(samplingRate)
FFTFreq = np.fft.rfftfreq(nPerSeg, 1/samplingRate)

if averaged:
    FFTMag = np.mean(FFTMag,axis=-1).flatten()
else:
    FFTMag = FFTMag[0,:,:]

#return FFTFreq.flatten(), FFTMag.real

fftFreq = FFTFreq.flatten()
fftMag = FFTMag.real

######################################################################

# Write the fftData to a .txt file in 2 columns
fftData = np.array([fftFreq, fftMag])
fftData = fftData.T

print(fftData)

filepath_3 = '/home/debian/my_data/fftData.txt'

with open(filepath_3, 'wb') as datafile_id:
    # The fftData values are rounde up to e-6
    np.savetxt(datafile_id, fftData, fmt=['%.8e','%.8e'])
```

A.3 Algorithm to Analyze the Spectrum

This algorithm analyzes the spectrum and provides feedback as to the bearing condition.

```
'''
# Algorithm for Analyzing the Spectrum automatically
# Created by Niklas Tritschler
# Copyright © 2019. All rights reserved.
'''

import numpy as np

filepath1 = '/home/debian/my_data/fftData.txt'
filepath2 = '/home/debian/my_data/condData.txt'

# Process the file so that you get the bearing characteristics,
# the RPM and the samplingRate

with open(filepath2, 'r') as f:
    lines = f.readlines()
    for x in lines:
        RPM = float(lines[0].strip("\n"))
        dP = float(lines[1].strip("\n"))
        dB = float(lines[2].strip("\n"))
        nB = float(lines[3].strip("\n"))
        angle = float(lines[4].strip("\n"))
        samplingRate = float(lines[5].strip("\n"))

# Reading the FFT Data file:

fftFreq = []
fftMag = []

with open(filepath1, 'r') as f:
    lines = f.readlines()
    for x in lines:
        fftFreq.append(float(x.split(' ')[0]))
        fftMag.append(float(x.split(' ')[1]))

# Splitting the FFT into different segments with
# deltaF = 2*Fspindle.
# This piece of code creates a three dimensional array that
# contains the segments of fftData.

numSeg = round((60*fftFreq[-1])/(2*RPM))
contSeg = round(len(fftFreq)/numSeg)

fftData = np.zeros((2, int(numSeg), int(contSeg)))

reps = 0

for i in range(0,numSeg):
```

```
    for j in range(0, contSeg):
        fftData[0][i][j] = fftFreq[j+reps*(contSeg-1)]
        fftData[1][i][j] = fftMag[j+reps*(contSeg-1)]
    reps += 1

# Now we have to find the peaks in each segment of the fftData:
# 1. Looking for the highest points of each segment
# 2. Checking if there are points on the left and right that
#    are +- 5 Hz away
# 3. If there are high points very close, delete these from the
#    list of highest points --> getting peaks

# 1.
def getHighestPoints(dataFrequency, dataMagnitude, numHighestPoints):

    ind = np.argpartition(dataMagnitude, -numHighestPoints)\
    [-numHighestPoints:]

    p = np.zeros((numHighestPoints, 2))
    for i in range(0, len(ind)):
        p[i][0] = dataMagnitude[ind[i]]
        p[i][1] = dataFrequency[ind[i]]
        #p[i][0] = fftMag[ind[i]]
        #p[i][1] = fftFreq[ind[i]]

    b = np.argsort(p[:,1], axis = 0)

    peaks = p[b]

    return peaks

highPoints = np.zeros((int(numSeg), 10, 2))

for i in range(0, numSeg):
        highPoints[i] = getHighestPoints(fftData[0][i], \
                                         fftData[1][i], 10)

# 2. + 3.
for i in range(0, numSeg):
    for j in range(0, 9):
        if abs(highPoints[i][j][1] - highPoints[i][j+1][1]) < 5:
            if highPoints[i][j][0] < highPoints[i][j+1][0]:
                highPoints[i][j][0] = 0
                highPoints[i][j][1] = 0
            else:
                highPoints[i][j+1][0] = 0
                highPoints[i][j+1][1] = 0

    b = np.argsort(highPoints[i][:,1], axis = 0)

    highPoints[i] = highPoints[i][b]

# Now we have to calculate the frequencies that we want to compare
# with the measured peaks to find accordan$

# Spindle frequencies
maxISpindle = int(round(2500*60/RPM))
spindleFreq = [i*RPM/60 for i in range(1, maxISpindle)]
```

```
# BPFI
maxIBPFI = int(round(2500*60/(nB/2*(1+dB/dP*np.cos(angle))*RPM)))
innerDefect = [i*nB/2*(1+dB/dP*np.cos(angle))*RPM/60 for i in \
               range(1, maxIBPFI)]

# BPFO
maxIBPFO = int(round(2500*60/(nB/2*(1-dB/dP*np.cos(angle))*RPM)))
outerDefect = [i*nB/2*(1-dB/dP*np.cos(angle))*RPM/60 for i in \
               range(1, maxIBPFO)]

# BSF
maxIBSF = int(round(2500*60/(dP/(2*dB)*(1-((dB/dP)**2)\
                                        *(np.cos(angle)**2))*RPM)))
ballDefect = [i*dP/(2*dB)*(1-((dB/dP)**2)*(np.cos(angle)**2))*RPM/60\
              for i in range(1, maxIBSF)]

# FTF
maxIFTF = int(round(2500*60/(0.5*(1-dB/dP*np.cos(angle))*RPM)))
cageDefect = [i*0.5*(1-dB/dP*np.cos(angle))*RPM/60 for i in \
              range(1, maxIFTF)]


# Now we can compare the calculated frequencies with the measured peaks

thresErr = 3.5

# Spindle frequencies

coincideSpindle = []

for i in range(0, numSeg, 1):
    for j in range(0, 10, 1):
        for n in range(0, len(spindleFreq), 1):
            err = 100*(spindleFreq[n]-highPoints[i][j][1])/spindleFreq[n]

            if abs(err) < thresErr:
                coincideSpindle.append(highPoints[i][j][1])

# BPFI frequencies

coincideBPFI = []

for i in range(0, numSeg, 1):
    for j in range(0, 10, 1):
        for n in range(0, len(innerDefect), 1):
            err = 100*(innerDefect[n]-highPoints[i][j][1])/innerDefect[n]

            if abs(err) < thresErr:
                coincideBPFI.append(highPoints[i][j][1])

# BPFO frequencies

coincideBPFO = []

for i in range(0, numSeg, 1):
    for j in range(0, 10, 1):
        for n in range(0, len(outerDefect), 1):
            err = 100*(outerDefect[n]-highPoints[i][j][1])/outerDefect[n]
```

```
            if abs(err) < thresErr:
                coincideBPFO.append(highPoints[i][j][1])

# BSF frequencies

coincideBSF = []

for i in range(0, numSeg, 1):
    for j in range(0, 10, 1):
        for n in range(0, len(ballDefect), 1):
            err = 100*(ballDefect[n]-highPoints[i][j][1])/ballDefect[n]

            if abs(err) < thresErr:
                coincideBSF.append(highPoints[i][j][1])

# Right now, I don´t look at FTF, since this never occurs

# We have to detect at least 3 harmonics to be able to detect a defect
# Therefore, we check if the coincide frequencies have 3 different
# frequencies (e.g. 1X, 2X, 3X)
# If not, then there is no defect

# The err can be +- thresErr --> Let us check if the coincide frequencies
# are in this range

for j in range(0, len(spindleFreq)):
    for i in range(0, len(coincideSpindle)):

        if (1-thresErr/100)*spindleFreq[j] <= coincideSpindle[i] <= \
        (1+thresErr/100)*spindleFreq[j]:
            coincideSpindle[i] = spindleFreq[j]

for i in range(0, len(coincideSpindle)-1, 1):

    if coincideSpindle[i] == coincideSpindle[i+1]:
        coincideSpindle[i] = 0

for j in range(0, len(innerDefect)):
    for i in range(0, len(coincideBPFI)):

        if (1-thresErr/100)*innerDefect[j] <= coincideBPFI[i] <= \
        (1+thresErr/100)*innerDefect[j]:
            coincideBPFI[i] = innerDefect[j]

for i in range(0, len(coincideBPFI)-1, 1):

    if coincideBPFI[i] == coincideBPFI[i+1]:
        coincideBPFI[i] = 0

for j in range(0, len(outerDefect)):
    for i in range(0, len(coincideBPFO)):

        if (1-thresErr/100)*outerDefect[j] <= coincideBPFO[i] <= \
        (1+thresErr/100)*outerDefect[j]:
            coincideBPFO[i] = outerDefect[j]

for i in range(0, len(coincideBPFO)-1, 1):

    if coincideBPFO[i] == coincideBPFO[i+1]:
```

```
        coincideBPFO[i] = 0

for j in range(0, len(ballDefect)):
    for i in range(0, len(coincideBSF)):

        if (1-thresErr/100)*ballDefect[j] <= coincideBSF[i] <= \
        (1+thresErr/100)*ballDefect[j]:
            coincideBSF[i] = ballDefect[j]

for i in range(0, len(coincideBSF)-1, 1):

    if coincideBSF[i] == coincideBSF[i+1]:
        coincideBSF[i] = 0

# If 3 numbers > 0 --> We detect the frequency

countSpindle = 0
countBPFI = 0
countBPFO = 0
countBSF = 0

for i in range(0, len(coincideSpindle)):
    if coincideSpindle[i] > 0.0:
        countSpindle += 1

for i in range(0, len(coincideBPFI)):
    if coincideBPFI[i] > 0.0:
        countBPFI += 1

for i in range(0, len(coincideBPFO)):
    if coincideBPFO[i] > 0.0:
        countBPFO += 1

for i in range(0, len(coincideBSF)):
    if coincideBSF[i] > 0.0:
        countBSF += 1

if countSpindle > 2:
    print("Detect Spindle Harmonics")

if countBPFI > 2:
    print("Detect BPFI Defect")

if countBPFO > 2:
    print("Detect BPFO Defect")

if countBSF > 2:
    print("Detect BSF Defect")
```

APPENDIX B. DESCRIPTION OF THE NODE-RED FLOW

The Node-Red flow is presented. With this flow, the vibration measurement, the transformation in the frequency domain and the analysis of the spectrum is initiated. Moreover, this flow publishes information about the bearing condition to the cloud.

B.1 Node-Red Flow

The Node-Red flow is presented in its JSON format.

[{"id":"8b31af3b.7e297","type":"tab","label":"Master´s Thesis","disabled":false,"info":""},{"id":"fab3a859.40b728","type":"inject","z":"8b31af3b.7e297","name":"Start Analysis","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":130,"y":340,"wires":[["6e8600b6.0929b","58ec0b9e.aea254"]]},{"id":"6e8600b6.0929b","type":"function","z":"8b31af3b.7e297","name":"Inject Accelerometer Message","func":"msg.payload = \"StartAccelerometer|\";\n\nreturn msg;","outputs":1,"noerr":0,"x":370,"y":380,"wires":[["13fafb10.8a38c5"]]},{"id":"13fafb10.8a38c5","type":"serial out","z":"8b31af3b.7e297","name":"Serial Out","serial":"a8b6d7c9.ea6318","x":580,"y":380,"wires":[]},{"id":"5df6d441.b62bac","type":"serial in","z":"8b31af3b.7e297","name":"Serial In","serial":"a8b6d7c9.ea6318","x":120,"y":460,"wires":[["8a640d89.45a79"]]},{"id":"8a640d89.45a79","type":"change","z":"8b31af3b.7e297","name":"","rules":[{"t":"set","p":"dateTime","pt":"msg","to":"$now()","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":330,"y":460,"wires":[["15bbef85.3cff5"]]},{"id":"15bbef85.3cff5","type":"switch","z":"8b31af3b.7e297","name":"","property":"payload","propertyType":"msg","rules":[{"t":"cont","v":"END","vt":"str"},{"t":"else"}],"checkall":"true","repair":false,"outputs":2,"x":510,"y":460,"wires":[["6b6a71ea.004da"],["bb5e161a.5b4d38"]]},{"id":"6b6a71ea.004da","type":"change","z":"8b31af3b.7e297","name":"","rules":[{"t":"set","p":"complete","pt":"msg","to":"true","tot":"bool"}],"action":"","property":"","from":"","to":"","reg":false,"x":690,"y":460,"wires":[["bb5e161a.5b4d38"]]},{"id":"bb5e161a.5b4d38","type":"join","z":"8b31af3b.7e297","name":"Join Separate Strings","mode":"custom","build":"string","property":"payload","propertyType":"msg","key":"topic","joiner":"","joinerType":"str","accumulate":false,"timeout":"","count":"","reduceRight":false,"reduceExp":"","reduceInit":"","reduceInitType":"","reduceFixup":"","x":200,"y":540,"wires":[["1fb28240.61783e"]]},{"id":"8eaaea.fb61b518","type":"debug","

z":"8b31af3b.7e297","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","x":1070,"y":540,"wires":[]},{"id":"d5851b3.fffd6e8","type":"file","z":"8b31af3b.7e297","name":"Store rawVibration","filename":"/home/debian/my_data/rawVibrationData.txt","appendNewline":false,"createDir":true,"overwriteFile":"true","x":670,"y":540,"wires":[["6c751623.639598"]]},{"id":"6c751623.639598","type":"change","z":"8b31af3b.7e297","name":"","rules":[{"t":"set","p":"payload","pt":"msg","to":"Stored rawVibration Data","tot":"str"}],"action":"","property":"","from":"","to":"","reg":false,"x":860,"y":540,"wires":[["8eaaea.fb61b518","5d835b0f.29fa44"]]},{"id":"1fb28240.61783e","type":"change","z":"8b31af3b.7e297","name":"","rules":[{"t":"change","p":"payload","pt":"msg","from":"\\n","fromt":"str","to":"","tot":"str"}],"action":"","property":"","from":"","to":"","reg":false,"x":440,"y":540,"wires":[["d5851b3.fffd6e8"]]},{"id":"5d835b0f.29fa44","type":"exec","z":"8b31af3b.7e297","command":"python3 /home/debian/my_python/createFFT.py","addpay":false,"append":"","useSpawn":"false","timer":"","oldrc":true,"name":"execute FFT","x":1070,"y":620,"wires":[[],["f7febd96.8d4b"],[]]},{"id":"f7febd96.8d4b","type":"exec","z":"8b31af3b.7e297","command":"python3 /home/debian/my_python/analyzeSpectrum.py","addpay":false,"append":"","useSpawn":"false","timer":"","oldrc":false,"name":"Spectrum Analysis","x":1250,"y":620,"wires":[["10164c5f.852ce4"],["10164c5f.852ce4"],["10164c5f.852ce4"]]},{"id":"58ec0b9e.aea254","type":"function","z":"8b31af3b.7e297","name":"Set Condition Data","func":"// Machine Parameter\nvar spindleSpeed;\n// Bearing Characteristics\nvar outerDiameterBearing;\nvar innerDiameterBearing;\nvar numberBallsBearing;\nvar contactAngleBearing;\n// Sampling Rate\nvar accelerometerSamplingRate;\n\n// Here adjust conditions\nflow.set(\"spindleSpeed\", 10000);\nflow.set(\"outerDiameter\", 60); \nflow.set(\"innerDiameter\", 11.906);\nflow.set(\"numBalls\", 9);\nflow.set(\"contactAngle\", 0);\n\nspindleSpeed = flow.get(\"spindleSpeed\");\nouterDiameterBearing = flow.get(\"outerDiameter\");\ninnerDiameterBearing = flow.get(\"innerDiameter\");\nnumberBallsBearing = flow.get(\"numBalls\");\ncontactAngleBearing = flow.get(\"contactAngle\")\naccelerometerSamplingRate = global.get(\"accelerometerSamplingRate\");\n\nmsg.payload = spindleSpeed + \"\\n\" + outerDiameterBearing + \"\\n\" + innerDiameterBearing + \"\\n\" + numberBallsBearing + \"\\n\" + contactAngleBearing + \"\\n\" + accelerometerSamplingRate + \"\\n\";\n\nreturn msg;","outputs":1,"noerr":0,"x":330,"y":300,"wires":[["e233e7ca.eba228"]]},{"id":"e233e7ca.eba228","type":"file","z":"8b31af3b.7e297","name":"Store Condition Data","filename":"/home/debian/my_data/condData.txt","appendNewline":false,"createDir":true,"overwriteFile":"true","x":540,"y":300,"wires":[["fb146e45.7f2fc"]]},{"id":"95eb06ee.9ffab8","type":"debug","z":"8b31af3b.7e297","name":"","active":true,"tosidebar":tr

ue,"console":false,"tostatus":false,"complete":"payload","x":910,"y":300,"wires":[]},{"id":"10164c5f.852ce4","type":"debug","z":"8b31af3b.7e297","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","x":1450,"y":620,"wires":[]},{"id":"fb146e45.7f2fc","type":"change","z":"8b31af3b.7e297","name":"","rules":[{"t":"set","p":"payload","pt":"msg","to":"Stored Condition Data","tot":"str"}],"action":"","property":"","from":"","to":"","reg":false,"x":740,"y":300,"wires":[["95eb06ee.9ffab8"]]},{"id":"3e2b7ac5.1eea26","type":"comment","z":"8b31af3b.7e297","name":"Instructions","info":"1. Set accelerometer properties\n2. Set the necessary condition data\n2. Start Analysis by clicking on blue button","x":110,"y":20,"wires":[]},{"id":"51dba08b.f685a","type":"function","z":"8b31af3b.7e297","name":"Set Accelerometer","func":"var accelerometerSamplingRate;\nvar accelerometerNumPoints;\n\nglobal.set(\"accelerometerSamplingRate\", 5250);\nglobal.set(\"accelerometerNumPoints\",5250);\n//4125\nglobal.set(\"accelerometerUnits\",\"Volts\");\nglobal.set(\"accelerometerPartId\",\"ADXL203EB\");\n\naccelerometerSamplingRate = global.get(\"accelerometerSamplingRate\");\naccelerometerNumPoints = global.get(\"accelerometerNumPoints\");\n\nmsg.payload = \"SetAccelerometer\" + \"|\" + accelerometerSamplingRate + \"|\" + accelerometerNumPoints;\n\nreturn msg;","outputs":1,"noerr":0,"x":470,"y":140,"wires":[["e5893f11.cba8d","d6b53a9.c2f50c8"]]},{"id":"3c8a3035.d2495","type":"inject","z":"8b31af3b.7e297","name":"Trigger Accelerometer Properties","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":true,"onceDelay":"0.1","x":200,"y":140,"wires":[["51dba08b.f685a"]]},{"id":"e5893f11.cba8d","type":"serial out","z":"8b31af3b.7e297","name":"Serial Out","serial":"a8b6d7c9.ea6318","x":680,"y":140,"wires":[]},{"id":"d6b53a9.c2f50c8","type":"debug","z":"8b31af3b.7e297","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","x":690,"y":80,"wires":[]},{"id":"4f55a950.200d58","type":"comment","z":"8b31af3b.7e297","name":"Set accelerometer variables.","info":"","x":160,"y":80,"wires":[]},{"id":"f69400a7.23ac4","type":"comment","z":"8b31af3b.7e297","name":"Set Condition Variables and Start Analysis","info":"","x":200,"y":220,"wires":[]},{"id":"a8b6d7c9.ea6318","type":"serial-port","z":"","serialport":"/dev/ttyS1","serialbaud":"115200","databits":"8","parity":"none","stopbits":"1","newline":"\\n","bin":"false","out":"char","addchar":false,"responsetimeout":"10000"}]

APPENDIX C. ERROR CALCULATION ON THE BRIDGEPORT V480

This appendix exemplarily illustrates the calculation of the error to show that only the peaks in the range of an error of up to 3.5 % are collected. The error calculation of one measurement on the CNC milling machine Bridgeport V480 is shown. Its spectrum is presented in Figure 4.7. During the measurement, the spindle had a frequency of 166.67 Hz. The BPFI was 898.82 Hz and the BSF was 403.42 Hz. The tables show the measured magnitude, the measured frequency, the harmonic, the calculated frequency and the calculated error. Only frequencies with an error smaller than 3.5 % are presented.

C.1 Error Calculation of the BPFI Harmonics

If the error is smaller than 3.5 %, the measured frequency is considered as a BPFI harmonic. The detected BPFI harmonics are shown in Table 4. The algorithm did not indicate a defect on the inner race, because less than three BPFI harmonics were detected.

Table 4: Detected BPFI Harmonics

Measured Magnitude	Measured Frequency [Hz]	BPFI Harmonic	Calculated Frequency [Hz]	Calculated Error [%]
3.84E-07	880.39	1	898.82	2.09
4.01E-08	1741.92	2	1797.64	3.10

C.2 Error Calculation of the Spindle Harmonics

If the error is smaller than 3.5 %, the measured frequency is considered as a spindle harmonic. The detected spindle harmonics are shown in Table 5.

Table 5: Detected Spindle Harmonics

Measured Magnitude	Measured Frequency [Hz]	Spindle Harmonic	Calculated Frequency [Hz]	Calculated Error [%]
5.03E-06	659.62	4	666.68	1.06
1.33E-07	1133.46	7	1166.67	2.85
8.26E-08	1324.62	8	1333.33	0.65
5.01E-08	1467.31	9	1500.00	2.18
9.38E-08	1620.77	10	1666.67	2.75
4.30E-08	1814.62	11	1833.33	1.02
4.74E-08	1965.39	12	2000.00	1.73
7.78E-08	2199.62	13	2166.67	1.52
4.80E-08	2342.31	14	2333.33	0.39

C.3 Error Calculation of the BSF Harmonics

If the error is smaller than 3.5 %, the measured frequency is considered as a BSF harmonic. The detected BSF harmonics are shown in Table 6. The algorithm indicated a rolling element defect, because three BSF harmonics were detected.

Table 6: Detected BSF Harmonics

Measured Magnitude	Measured Frequency [Hz]	BSF Harmonic	Calculated Frequency [Hz]	Calculated Error [%]
1.66E-07	409.23	1	403.42	1.44
1.98E-06	780.77	2	806.84	3.23
4.92E-08	1225.00	3	1210.26	1.22
1.11E-07	1599.23	4	1613.68	0.90
9.89E-08	1984.23	5	2017.10	1.63

REFERENCES

[1] Brüel & Kjaer, Machine-condition monitoring using vibration analysis – permanent monitoring of an Austrian paper mill, 1988.

[2] G. Fliedner, Leading and managing the lean management process, firstst ed., Business Expert Press, [New York, N.Y.], 2012.

[3] Information on https://www2.deloitte.com/insights/us/en/focus/industry-4-0/using-predictive-technologies-for-asset-maintenance.html

[4] R. Ahmad, S. Kamaruddin, An overview of time-based and condition-based maintenance in industrial application, Computers & Industrial Engineering 63 (2012) 135–149.

[5] R.B. Randall, Vibration-based condition monitoring: Industrial, aerospace, and automotive applications, Wiley, Chichester, West Sussex, U.K, Hoboken, NJ, 2011.

[6] N.I. Azeez, A.C. Alex, Detection of rolling element bearing defects by vibration signature analysis: A review, in: Annual International Conference on Emerging Research Areas: Magnetics, Machines and Drives (AICERA/iCMMD), 2014: 24 - 26 July 2014, Amal Jyothi College of Engineering, Kanjirappally, Kottayam, Kerala, India ; proceedings, IEEE, Piscataway, NJ, 2014, pp. 1–5.

[7] G.S. Maruthi, V. Hegde, Application of MEMS Accelerometer for Detection and Diagnosis of Multiple Faults in the Roller Element Bearings of Three Phase Induction Motor, IEEE Sensors J. 16 (2016) 145–152.

[8] L.M. Simon (Ed.), Fault detection: Theory, methods and systems, Nova Science Publishers, New York, 2011.

[9] B. Sreejith, A.K. Verma, A. Srividya, Fault diagnosis of rolling element bearing using time-domain features and neural networks, in: 2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems, IEEE, 2008, pp. 1–6.

[10] M. Firla, Z.-Y. Li, N. Martin, C. Pachaud, T. Barszcz, Automatic characteristic frequency association and all-sideband demodulation for the detection of a bearing fault, Mechanical Systems and Signal Processing 80 (2016) 335–348.

[11] Information on https://www.emeraldinsight.com/doi/full/10.1108/13552510110404512

[12] Z. Kıral, H. Karagülle, Vibration analysis of rolling element bearings with various defects under the action of an unbalanced force, Mechanical Systems and Signal Processing 20 (2006) 1967–1991.

[13] T.A. Harris, M.N. Kotzalas, Essential concepts of bearing technology, fifth. ed., CRC Press, Boca Raton, Fla., 2007.

[14] T.A. Harris, Essential Concepts of Bearing Technology, FIFTH EDITION, CRC Press, 2006.

[15] X. Tian, J. Xi Gu, I. Rehab, G.M. Abdalla, F. Gu, A.D. Ball, A robust detector for rolling element bearing condition monitoring based on the modulation signal

bispectrum and its performance evaluation against the Kurtogram, Mechanical Systems and Signal Processing 100 (2018) 167–187.

[16] Information on https://power-mi.com/content/typical-bearing-defects-and-spectral-identification

[17] Information on https://www.mobiusinstitute.com/site2/item.asp?LinkID=8011&iVibe=1&sTitle=Rolling%20element%20bearings

[18] L.D. Meyer, F.F. Ahlgren, B. Weichbrodt, An Analytic Model for Ball Bearing Vibrations to Predict Vibration Response to Distributed Defects, J. Mech. Des. 102 (1980) 205.

[19] T.E. Tallian, O.G. Gustafsson, Progress in Rolling Bearing Vibration Research and Control, A S L E Transactions 8 (1965) 195–207.

[20] C.S. Sunnersjö, Rolling bearing vibrations—The effects of geometrical imperfections and wear, Journal of Sound and Vibration 98 (1985) 455–474.

[21] N. Tandon, A. Choudhury, A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings, Tribology International 32 (1999) 469–480.

[22] T. Igarashi, H. Hamada, Studies on the Vibration and Sound of Defective Rolling Bearings: First Report Vibration of Ball Bearings with One Defect, Bulletin of JSME 25 (1982) 994–1001.

[23] T.A. Harris, Rolling bearing analysis, fourth. ed., Wiley, New York, NY, 2001.

[24] B. Dolenc, P. Boškoski, Đ. Juričić, Distributed bearing fault diagnosis based on vibration analysis, Mechanical Systems and Signal Processing 66-67 (2016) 521–532.

[25] S. Mokhatab (Ed.), Handbook of natural gas transmission and processing: Principles and practices, Fourth edition, Gulf Professional Publishing, Oxford, 2019.

[26] M.P. Boyce (Ed.), Gas turbine engineering handbook, fourth. ed., Butterworth-Heinemann; Elsevier, Waltham, Mass, Amsterdam, 2012.

[27] Y. Shao, F. Nezu, Bearing fault detection using laser displacement sensor, in: 1996 Annual Conference of the Society of Instrument and Control (SICE), IEEE, Piscataway, 1996, pp. 1069–1072.

[28] M.P. Boyce, Control Systems and Instrumentation, in: M.P. Boyce (Ed.), Gas turbine engineering handbook, fourth. ed., Butterworth-Heinemann; Elsevier, Waltham, Mass, Amsterdam, 2012, pp. 721–768.

[29] B. Sun, B. Li, Laser Displacement Sensor in the Application of Aero-Engine Blade Measurement, IEEE Sensors J. 16 (2016) 1377–1384.

[30] J. Rejc, J. Činkelj, M. Munih, Dimensional measurements of a gray-iron object using a robot and a laser displacement sensor, Robotics and Computer-Integrated Manufacturing 25 (2009) 155–167.

[31] A. Sabato, C. Niezrecki, G. Fortino, Wireless MEMS-Based Accelerometer Sensor Boards for Structural Vibration Monitoring: A Review, IEEE Sensors J. 17 (2017) 226–235.

[32] Matej Andrejasic, MEMS Accelerometers.

[33] MTS Systems Corporation, Product Specification Sheet: Low Frequency Industrial ICP Accelerometer, Depew, NY, 2019.

[34] Analog Devices Inc., Data Sheet: ADXL103/ADXL203, Norwood, MA, USA, 2019.

[35] J. Laine, D. Mougenot, A high-sensitivity MEMS-based accelerometer, The Leading Edge 33 (2014) 1234–1242.

[36] H. Austerlitz, Analog/Digital Conversions, in: H. Austerlitz (Ed.), Data acquisition techniques using PCs, secondnd ed., Academic Press, Amsterdam, Boston, 2003, pp. 51–77.

[37] A. Brandt, K. Ahlin, Sampling and Time-Domain Analysis, Sound & Vibration 05/2010 (2010).

[38] E.P. Carden, P. Fanning, Vibration Based Condition Monitoring: A Review, Structural Health Monitoring 3 (2016) 355–377.

[39] J. Mathew, R.J. Alfredson, The Condition Monitoring of Rolling Element Bearings Using Vibration Analysis, J. Vib. Acoust. 106 (1984) 447.

[40] O.R. Seryasat, M. Aliyari shoorehdeli, F. Honarvar, A. Rahmani, Multi-fault diagnosis of ball bearing using FFT, wavelet energy entropy mean and root mean square (RMS), in: 2010 IEEE International Conference on Systems, Man and Cybernetics: SMC 2010 October 10-13, 2010, Istanbul, Turkey, IIEEE, Piscataway, 2010, pp. 4295–4299.

[41] T. WILLIAMS, X. RIBADENEIRA, S. BILLINGTON, T. KURFESS, ROLLING ELEMENT BEARING DIAGNOSTICS IN RUN-TO-FAILURE LIFETIME TESTING, Mechanical Systems and Signal Processing 15 (2001) 979–993.

[42] N. Tandon, A comparison of some vibration parameters for the condition monitoring of rolling element bearings, Measurement 12 (1994) 285–289.

[43] R.B.W. Heng, M.J.M. Nor, Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition, Applied Acoustics 53 (1998) 211–226.

[44] M. Christman, Bearing Fault Detection PLUS - Definition of Variables, New York.

[45] Z. Kiral, H. Karagülle, Simulation and analysis of vibration signals generated by rolling element bearing with defects, Tribology International 36 (2003) 667–678.

[46] G.D. Bergland, A guided tour of the fast Fourier transform, IEEE Spectr. 6 (1969) 41–52.

[47] W.T. Cochran, J.W. Cooley, D.L. Favin, H.D. Helms, R.A. Kaenel, W.W. Lang, G.C. Maling, D.E. Nelson, C.M. Rader, P.D. Welch, What is the fast Fourier transform?, Proc. IEEE 55 (1967) 1664–1674.

[48] H. Sorensen, D. Jones, M. Heideman, C. Burrus, Real-valued fast Fourier transform algorithms, IEEE Trans. Acoust., Speech, Signal Process. 35 (1987) 849–863.

[49] H. Wen, Z. Teng, S. Guo, J. Wang, B. Yang, Y. Wang, T. Chen, Hanning self-convolution window and its application to harmonic analysis, Sci. China Ser. E-Technol. Sci. 52 (2009) 467–476.

[50] F. Zhang, Z. Geng, W. Yuan, The algorithm of interpolating windowed FFT for harmonic analysis of electric power system, IEEE Trans. Power Delivery 16 (2001) 160–164.

[51] M. Cerna, A.F. Harvey, The Fundamentals of FFT-Based Signal Analysis and Measurement, Conference Proceedings (1993).

[52] K.F. Chen, Y.F. Li, Combining the Hanning windowed interpolated FFT in both directions, Computer Physics Communications 178 (2008) 924–928.

[53] S. Rapuano, F. Harris, An introduction to FFT and time domain windows, IEEE Instrum. Meas. Mag. 10 (2007) 32–44.

[54] P. Jayaswal, A.K. Wadhwani, K.B. Mulchandani, Machine Fault Signature Analysis, International Journal of Rotating Machinery 2008 (2008) 1–10.

[55] N.G. Nikolaou, I.A. Antoniadis, Rolling element bearing fault diagnosis using wavelet packets, NDT & E International 35 (2002) 197–205.

[56] J.R. BLOUGH, DEVELOPMENT AND ANALYSIS OF TIME VARIANT DISCRETE FOURIER TRANSFORM ORDER TRACKING, Mechanical Systems and Signal Processing 17 (2003) 1185–1199.

[57] F. Al-Badour, M. Sunar, L. Cheded, Vibration analysis of rotating machinery using time–frequency analysis and wavelet techniques, Mechanical Systems and Signal Processing 25 (2011) 2083–2101.

[58] R.M. White, A Sensor Classification Scheme, IEEE Trans. Ultrason., Ferroelect., Freq. Contr. 34 (1987) 124–126.

[59] D.E. Bolanakis, Microcontroller Education: Do it Yourself, Reinvent the Wheel, Code to Learn, Synthesis Lectures on Mechanical Engineering 1 (2017) 1–193.

[60] W.P. Birmingham, D.P. Siewiorek, MICON: A Knowledge Based Single Board Computer Designer, in: P.H. Lambert (Ed.), Proceedings of the 21st Design Automation Conference, IEEE Press, Piscataway, NJ, 1984, pp. 565–571.

[61] Information on https://www.python.org/about/apps/

[62] P.T. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2003) 114–131.

[63] U. Hunkeler, H.L. Truong, A. Stanford-Clark, MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks, in: 3rd International Conference on

Communication Systems Software and Middleware and workshops, 2008: COMSWARE 2008 ; 6 - 10 Jan. 2008, Bangalore, India, IEEE, Piscataway, NJ, 2008, pp. 791–798.

[64] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C.K.-Y. Tan, Performance evaluation of MQTT and CoAP via a common middleware, in: IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014: 21 - 24 April 2014, Singapore, IEEE, Piscataway, NJ, 2014, pp. 1–6.

[65] Information on https://nodered.org/

[66] Analog Devices Inc., Data Sheet: ADXL103/ADXL203, 2018.

[67] Analog Devices Inc., Data Sheet: ADXL1001/ADXL1002, 2017.

[68] Information on https://www.pjrc.com/teensy/teensy31.html

[69] Information on https://store.particle.io/products/photon

[70] I. Particle Industries, Particle Photon: Datasheet.

[71] Information on https://www.emco-world.com/en/products/industry/milling/cat/27/d/2/p/1000236%2C27/pr/emcomill-e350.html

[72] Information on https://www.hardinge.com/product/milling/v-series/#carouselExampleControls